KLASIFIKASI STATUS GIZI PADA BALITA MENGGUNAKAN METODE *EXTREME LEARNING MACHINE* DAN ALGORITME GENETIKA

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Nabila Lubna Irbakanisa
NIM: 155150207111016

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

PENGESAHAN

KLASIFIKASI STATUS GIZI PADA BALITA MENGGUNAKAN
EXTREME LEARNING MACHINE DAN
ALGORITME GENETIKA

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh:
Nabila Lubna Irbakanisa
NIM: 155150207111016

Skripsi ini telah diuji dan dinyatakan lulus pada
3 Januari 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Imam Cholissodin, S.Si, M. Kom	Fitra A. Bachtiar, Dr.Eng, S.T, M.Eng
NIK: 201201 850719 1 001	NIK: 201201 840628 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 17 Desember 2018

Nabila Lubna Irbakanisa

NIM: 155150207111016

KATA PENGANTAR

Puji dan syukur atas kehadirat Allah SWT karena atas rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi ini dengan tepat waktu. Selama pengerjaan skripsi penulis mendapatkan ilmu pengetahuan serta dorongan dan masukan dari berbagai pihak. Maka pada kesempatan ini, perkenankanlah penulis mengucapkan terima kasih kepada:

1. Bapak Imam Cholissodin, S.Si, M.Kom selaku Dosen Pembimbing I yang telah bersedia meluangkan waktunya untuk membimbing penulis dengan penuh kesabaran dalam menyelesaikan skripsi ini.
2. Bapak Fitra Abdurrachman Bachtiar, Dr.Eng, S.T, M.Eng selaku Dosen Pembimbing II yang telah bersedia meluangkan waktunya untuk membimbing penulis dengan penuh kesabaran dalam menyelesaikan skripsi ini.
3. Bapak Adam Hendra Brata, S.Kom, M.T selaku Dosen Penasehat Akademik yang selalu memberikan nasehat kepada penulis selama menempuh masa studi.
4. Orang tua dan seluruh keluarga atas segenap kasih sayang, doa, dan dukungan yang telah diberikan.
5. Mokhamad Zukhruf Mifta Al Firdaus yang selalu memberikan dukungan, bantuan, dan motivasi kepada penulis.
6. Teman-teman Informatika, khususnya Akhmad Rohim dan Habridio Kurniawan Putra yang telah memberikan saran, kritik, bantuan, serta motivasi kepada penulis.
7. Teman-teman Lembaga Pers Mahasiswa DISPLAY yang telah memberikan dukungan dan motivasi kepada penulis.
8. Serta semua pihak yang telah membantu penulis secara langsung maupun tidak langsung dalam menyelesaian skripsi ini.

Dalam penyusunannya, skripsi ini tentu masih jauh dari sempurna. Untuk itu, penulis mengharapkan kritik dan saran yang bersifat membangun dari semua pihak demi menyempurnakan skripsi ini.

Malang, 17 Desember 2018

Penulis

nabilalubna@student.ub.ac.id

ABSTRAK

Nabila Lubna Irbakanisa, Klasifikasi Status Gizi pada Balita Menggunakan Metode *Extreme Learning Machine* dan Algoritme Genetika

Pembimbing: Imam Cholissodin, S.Si, M.Kom. dan Fitra Abdurrachman Bachtiar, Dr.Eng., S.T, M.Eng.

Permasalahan gizi merupakan salah satu masalah yang serius. Karena gizi tidak hanya menyangkut tentang kelangsungan hidup, melainkan berkaitan juga dengan kualitas hidup seseorang. Dalam kasus ini, pemeriksaan status gizi yang dilakukan oleh tenaga medis umumnya dilakukan dengan pengarsipan, yaitu dengan melakukan pencatatan secara manual dan kemudian dianalisis dengan membandingkan data pasien dengan standar baku status gizi. Namun, dengan melakukan analisis secara manual tersebut membuat rentannya ketidaktelitian dalam pengidentifikasian status gizi, serta memakan waktu lebih lama dikarenakan kurang praktis. Berdasarkan masalah tersebut, penulis menerapkan metode *Extreme Learning Machine* (ELM) dan Algoritme Genetika untuk mengklasifikasikan status gizi pada balita secara cepat dan akurat. Pada penelitian ini, Algoritme Genetika berperan untuk mencari bobot awal terbaik, yang selanjutnya akan digunakan untuk penentuan nilai status gizi menggunakan ELM. Setelah dilakukan pengujian, diperoleh rata-rata akurasi oleh metode ELM – Algoritme Genetika sebesar 72.3529% dengan parameter *popsize* bernilai 100, iterasi sebanyak 34, nilai *cr* sebesar 0.6, nilai *mr* sebesar 0.4, dan *hidden neuron* sebanyak 2. Sedangkan akurasi yang didapatkan dari algoritme ELM biasa adalah 67.6471%. Dari hasil akurasi yang didapatkan menunjukkan bahwa penggabungan Algoritme Genetika pada ELM mampu meningkatkan akurasi.

Kata kunci: Status Gizi, *Extreme Learning Machine*, Jaringan Syaraf Tiruan, Algoritme Genetika

ABSTRACT

Nabila Lubna Irbakanisa, Classification Nutritional Status in Toddlers Using Extreme Learning Machine and Genetic Algorithm

Supervisors: Imam Cholissodin, S.Si, M.Kom and Fitra Abdurrachman Bachtiar, Dr.Eng, S.T, M.Eng

Nutritional problem is one of serious problems. Because nutrition does not only concern in survival, but also relates to the quality of someone's life. In this case, the examination of child nutrient by medical personnel is generally done by archiving, namely by recording manually, and then analyzed by comparing patient's data with the standard of nutritional status. But by doing the analysis manually, it makes the vulnerability of inaccuracy in identifying nutritional status, and takes longer time because it is less practical. Based on these problems, the authors apply the Extreme Learning Machine (ELM) method and Genetic Algorithm to classify nutritional status in toddlers quickly and accurately. In this research, Genetic Algorithms used for finding the best input weight, which will then be used to determine the value of nutritional status using ELM. After testing, obtained an average accuracy of ELM – Genetic Algorithm is 72.3529% with the number of popsize is 100, 34 iterations, crossover rate 0.6, mutation rate 0.4, and 2 hidden neuron. While the accuracy obtained from the ELM is 67.6471%. The result shows that the addition if Genetic Algorithm on ELM can improve the accuracy.

Keywords: *Nutritional Status, Extreme Learning Machine, Artificial Neural Network, Genetic Algorithm*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Kesehatan merupakan hal yang sangat penting bagi setiap individu di suatu negara, sebagaimana tercantum dalam Deklarasi Hak Asasi Manusia (HAM) oleh Perserikatan Bangsa-Bangsa (PBB) pada tanggal 10 November 1948 yang menyatakan bahwa setiap manusia memiliki hak atas standar hidup yang baik dan layak bagi kesehatan dan kesejahteraan diri sendiri dan keluarganya. Karena tanpa adanya kesehatan, manusia tidak dapat melakukan segala sesuatu dengan optimal. Bahkan dalam lingkup besar secara tidak langsung dapat menyebabkan penurunan kualitas penduduk di suatu negara apabila negara tersebut melakukan penanganan dengan lambat.

Terdapat banyak macam masalah kesehatan, salah satunya adalah masalah gizi. Anak usia bawah lima tahun (balita) rentan terhadap masalah kesehatan dan gizi (Irsyadi & Fathina, 2015). Berdasarkan Pantauan Status Gizi (PSG) 2017 oleh Kementerian Kesehatan Republik Indonesia, balita yang memiliki permasalahan gizi pada tahun 2017 mencapai 17,8%, yang terdiri dari 3,8% menderita gizi buruk dan 14% mengalami gizi kurang. Permasalahan gizi ini bukanlah sesuatu yang kecil, karena gizi tidak hanya menyangkut tentang kelangsungan hidup melainkan berkaitan juga dengan kualitas hidup seseorang. Masalah gizi dapat terjadi di segala usia, dimulai saat masih ada di dalam kandungan, bayi, anak-anak, remaja, dewasa, bahkan lanjut usia. Kehidupan di dua tahun pertama merupakan masa yang penting, karena masa tersebut memiliki probabilitas utama untuk mencegah gangguan pertumbuhan dan perkembangan. Pada periode tersebut juga akan menimbulkan gangguan gizi yang bersifat permanen, tidak dapat dipulihkan meskipun kebutuhan gizi pada masa selanjutnya tercukupi (De Onis, Blössner, & Borghi, 2012). Selain itu, masalah gizi pada balita tidak mudah dikenali oleh pemerintah atau masyarakat, bahkan keluarga. Akibatnya, bila terdapat sejumlah anak pada suatu desa yang memiliki masalah gizi, mereka tidak segera mendapat perhatian dikarenakan anak tersebut terkadang tidak tampak sakit (Irsyadi & Fathina, 2015).

Berdasarkan hasil wawancara yang dilakukan penulis kepada petugas gizi di Puskesmas Pandanwangi Kota Malang, pemeriksaan status gizi yang dilakukan di tempat pelayanan kesehatan seperti Pusat Pelayanan Terpadu (Posyandu) umumnya dilakukan dengan pengarsipan, yaitu dengan melakukan pencatatan secara manual dan kemudian dianalisis dengan membandingkan data pasien dengan standar baku status gizi. Namun, dengan melakukan analisis secara manual tersebut membuat rentannya ketidaktelitian dalam pengidentifikasian status gizi, serta memakan waktu lebih lama dikarenakan kurang praktis. Oleh karena itu, perlunya sebuah aplikasi yang dapat mengklasifikasi status gizi pada balita yang dapat berguna untuk memberikan penilaian status gizi secara cepat dan akurat.

Terdapat banyak metode untuk menyelesaikan permasalahan, salah satunya yaitu menggunakan metode dari Jaringan Syaraf Tiruan (JST). JST merupakan model kecerdasan yang mengadopsi sistem pembelajaran pada otak manusia karena ia mampu menyimpan dan melakukan pembelajaran dari pengetahuan dan pengalaman layaknya otak manusia. Salah satu metode dari JST adalah *Extreme Learning Machine* (ELM). ELM merupakan metode baru dari JST yang biasa digunakan untuk melakukan klasifikasi, peramalan, dan lain-lain. ELM mampu memberikan tingkat akurasi dan kecepatan yang lebih baik dibandingkan dengan metode pembelajaran lainnya (Hidayat & Suprapto, 2012).

Penelitian tentang klasifikasi status gizi telah banyak dilakukan, salah satunya dilakukan oleh Anggraeni & Indrarti (2010). Penelitian tersebut menerapkan JST menggunakan *Backpropagation* untuk mengklasifikasikan status gizi balita ke dalam 4 kelas. Hasilnya, JST dapat diterapkan dengan akurasi 93,85%. Penelitian lain tentang metode ELM juga telah diimplementasikan ke berbagai permasalahan, seperti pengklasifikasian penyakit *chronic kidney disease* yang dilakukan oleh Fadilla, Adikara, dan Perdana (2018). Dengan menggunakan *hidden neuron* sebanyak 50, didapatkan hasil akurasi klasifikasi sebesar 96,7%. Untuk meningkatkan kinerja dari ELM, diperlukan metode lain yang dapat digabungkan dengan ELM. Menurut Ali, Zolkipli, Mohammed, & Jaber (2017) penentuan parameter secara acak pada ELM dapat menghasilkan parameter yang tidak optimal, sehingga akan berpengaruh pada kinerja dan stabilitas generalisasi. Oleh karena itu, pada penelitiannya tersebut diusulkan Algoritme Genetika untuk mengoptimalkan nilai bobot awal dan bias. Hasil dari penelitian tersebut adalah penggabungan metode ELM dan Algoritme Genetika memiliki kinerja yang baik karena mampu meningkatkan akurasi klasifikasi. Chandra, Santoso, dan Adinugroho (2018) juga telah menggunakan Algoritme Genetika dan ELM untuk menentukan kualitas air sungai. Peneliti menggunakan Algoritme Genetika untuk melakukan optimasi bobot awal yang digunakan pada proses pelatihan dan pengujian metode ELM. Hasil dari penelitian tersebut menunjukkan bahwa Algoritme Genetika mampu meningkatkan akurasi sebesar 0,1102% dari penelitian sebelumnya yang dilakukan oleh Azizah (2016).

Pada penelitian ini, penulis akan mengklasifikasikan status gizi pada balita menggunakan ELM dan Algoritme Genetika. Karena pembangkitan bobot yang secara acak pada ELM tidak menjamin bahwa bobot yang dibangkitkan adalah bobot yang optimal, maka Algoritme Genetika berperan untuk mencari bobot awal terbaik, yang selanjutnya akan digunakan untuk penentuan nilai status gizi menggunakan ELM, sehingga didapatkan parameter dan akurasi terbaik. Digunakannya Algoritme Genetika untuk melakukan pengoptimalan bobot dikarenakan algoritme ini dapat menyelesaikan permasalahan yang kompleks serta cukup fleksibel untuk dihibridasikan dengan algoritme lainnya (Wayan Firdaus Mahmudy, 2015).

Berdasarkan uraian tersebut, maka penulis memutuskan untuk mengajukan penelitian dengan judul "Klasifikasi Status Gizi pada Balita Menggunakan *Extreme Learning Machine* dan Algoritme Genetika" yang bertujuan untuk memudahkan

memudahkan petugas medis dalam mengklasifikasikan status gizi pada balita dengan cepat dan akurat.

1.2 Rumusan Masalah

Berdasarkan uraian yang terdapat pada latar belakang, maka didapatkan rumusan masalah sebagai berikut:

1. Bagaimana parameter yang optimal dari metode *Extreme Learning Machine* dan Algoritme Genetika untuk klasifikasi status gizi pada balita?
2. Bagaimana tingkat akurasi sistem klasifikasi status gizi pada balita menggunakan metode *Extreme Learning Machine* dan Algoritme Genetika?

1.3 Tujuan

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengetahui parameter yang optimal dari metode *Extreme Learning Machine* dan Algoritme Genetika untuk klasifikasi status gizi pada balita.
2. Mengetahui tingkat akurasi sistem klasifikasi status gizi pada balita.

1.4 Manfaat

Manfaat dari hasil penelitian ini yaitu:

1. Mendapatkan hasil yang cepat dan akurat untuk mengklasifikasikan status gizi pada balita menggunakan *Extreme Learning Machine* dan Algoritme Genetika.
2. Memudahkan petugas medis dalam mengklasifikasikan status gizi pada balita.

1.5 Batasan Masalah

Terdapat beberapa batasan permasalahan dari penelitian ini, yaitu:

1. Data yang digunakan berasal dari data rekam medis status gizi pada balita di Puskesmas Pandanwangi, Kota Malang.
2. Parameter yang digunakan pada penelitian ini meliputi jenis kelamin, umur (bulan), berat badan (kg), dan tinggi badan (cm).
3. Keluaran sistem berupa hasil status gizi pada balita menurut berat badan/umur (BB/U), tinggi badan/umur (TB/U), dan berat badan/tinggi badan (BB/TB).

1.6 Sistematika Pembahasan

Untuk mencapai tujuan yang diharapkan, maka akan dijelaskan secara garis besar isi dari bab-bab yang ada di dalam skripsi ini dengan sistematika penulisan sebagai berikut:

Bab 1 Pendahuluan

Bab pendahuluan menjelaskan tentang latar belakang dari permasalahan, perumusan masalah, tujuan dari rumusan masalah, manfaat penelitian, batasan masalah yang digunakan dalam penelitian, serta sistematika penulisan yang digunakan.

Bab 2 Landasan Kepustakaan

Bab landasan kepustakaan membahas tentang teori-teori yang relevan dalam penyelesaian penelitian ini, seperti penjelasan tentang status gizi, dan dasar teori dari *Extreme Learning Machine* serta Algoritme Genetika.

Bab 3 Metodologi

Bab metodologi menjelaskan tentang langkah-langkah atau metode penelitian yang digunakan untuk membuat sistem klasifikasi status gizi pada balita dan langkah kerja yang dilakukan dalam pembuatan sistem.

Bab 4 Perancangan

Bab perancangan membahas tentang langkah-langkah yang digunakan sebagai acuan dalam melakukan perancangan sistem klasifikasi status gizi pada balita menggunakan *Extreme Learning Machine* dan Algoritme Genetika.

Bab 5 Implementasi

Bab implementasi membahas tentang implementasi dari sistem yang dibangun berdasarkan hasil perancangan sistem yang telah dibuat sebelumnya.

Bab 6 Pengujian dan Analisis

Pada bab ini berisi pembahasan tentang alur pengujian serta analisis terhadap sistem yang telah dibangun.

Bab 7 Penutup

Bab penutup berisi kesimpulan dari penelitian yang telah dilakukan, serta saran untuk penelitian berikutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan terdiri atas kajian pustaka dan dasar teori yang berhubungan dengan judul. Kajian pustaka merupakan daftar referensi seperti buku, jurnal, hasil konferensi, atau sumber lain yang dijadikan penulis sebagai rujukan atau perbandingan terhadap penelitian yang dilakukan. Dasar teori merupakan landasan atau pondasi dalam penelitian, berisi teori-teori yang mendukung dalam pembuatan sistem.

2.1 Kajian Pustaka

Kajian pustaka dilakukan dengan cara melakukan analisis perbandingan terhadap penelitian-penelitian yang telah ada terkait sistem klasifikasi menggunakan objek dan metode yang sama dengan penelitian yang diusulkan. Analisis perbandingan beberapa penelitian terkait sistem Klasifikasi Status Gizi Pada Balita Menggunakan *Extreme Learning Machine* Dan Algoritme Genetika dapat dilihat pada Tabel 2.1.

Penelitian yang dilakukan oleh Anggraeni & Indrarti (2010) bertujuan untuk membangun model Jaringan Syaraf Tiruan untuk mengklasifikasikan status gizi balita ke dalam 4 kelas, yaitu gizi buruk, gizi kurang, gizi baik, dan gizi lebih menggunakan algoritme *Backpropagation*. Parameter yang digunakan adalah jenis kelamin, berat badan, penyakit, umur, dan status ekonomi. Sistem yang dibangun dengan 8 lapisan tersembunyi, *learning rate* 0.9, maksimum *epoch* 2000, dan momentum 0,3 memperoleh hasil akurasi sebesar 93,85%. Hasil tersebut menunjukkan bahwa JST dapat diterapkan pada klasifikasi gizi balita dengan baik.

Penelitian kedua merupakan penelitian yang dilakukan oleh Irsyadi & Fathina (2015). Penelitian tersebut bertujuan untuk mempermudah tenaga medis dalam memberikan penilaian terhadap status gizi balita yang berjenis kelamin laki-laki. Variabel yang digunakan dalam penelitiannya yaitu berat badan menurut umur (BB/U), tinggi badan menurut umur (TB/U), dan berat badan menurut tinggi badan (BB/TB). Dengan menggunakan *Backpropogation* sebagai metodenya, sistem mampu memperoleh akurasi sebesar 100% dengan konfigurasi jaringan 3 *input layer*, 3 *hidden neuron*, dan 1 *output layer*.

Penelitian berikutnya merupakan penelitian yang dilakukan oleh Amalia (2015) untuk memprediksi penyakit diabetes. Data yang digunakan sebanyak 768 data dengan 8 parameter, yaitu berapa kali hamil, tekanan darah, serum insulin, massa tubuh, diabetes sisilah, umur, konsentrasi glukosa, dan lipatan kulit. Tujuan dari penelitian ini yaitu mengolah dataset diabetes menggunakan metode *Backpropagation*, kemudian meningkatkan akurasinya menggunakan metode optimasi Algoritme Genetika. Hasil dari penelitian tersebut yaitu akurasi terbaik sebesar 74,46% yang diperoleh dari hasil momentum 0,3, *learning rate* 2, dan 5 *hidden neuron*. Sementara untuk pengolahan dataset diabetes menggunakan optimasi Algoritme Genetika mendapatkan akurasi sebesar 77,10% yang

diperoleh dari nilai momentum 0,3, *learning rate* 2, dan 2 *hidden neuron*. Penelitian ini juga menunjukkan bahwa Algoritme Genetika mampu meningkatkan akurasi dari metode *Neural Network*.

Penelitian keempat yaitu penelitian yang dilakukan oleh Fadilla, Adikara, dan Perdana (2018) yang bertujuan untuk mengklasifikasikan penyakit *Chronic Kidney Disease* (CKD). Sistem yang dibangun memiliki 24 parameter data, yaitu *age, white blood cell count, red blood cells, pus cell, specific gravity, albumin, sugar, bacteria, blood urea, serum creatinine, sodium, potassium, hemoglobin, blood pressure, packed cell volume, red blood cell count, blood glucose random, appetite, hypertension, diabetes mellitus, pedal edema,* anemia, dan *coronary artery disease.* Hasil penelitian menunjukkan bahwa dengan metode *Extreme Learning Machine*, sistem menunjukkan kinerja terbaik yaitu pada 50 *hidden neuron* serta dengan perbandingan 70:30 untuk data latih dan data uji.

Penelitian kelima yaitu penelitian yang dilakukan oleh Chandra, Santoso, dan Adinugroho (2018). Pada penelitian tersebut digunakan data pelatihan dan pengujian yang ditentukan menggunakan *k-fold cross validation* dengan k = 5 yang dibentuk dari 150 data awal. Hasil dari penelitian ini yaitu metode ELM dan Algoritme Genetika dapat diterapkan pada penentuan kualitas air sungai dengan akurasi 88,0002%, mengalami peningkatan sebesar 0,01102% dari penelitian yang dilakukan oleh Azizah (2016) yang sebelumnya memiliki akurasi 87,97%.

Penelitian keenam yaitu penelitian yang dilakukan oleh Ariestyani, Adikara, dan Perdana (2018) dengan tujuan untuk mengklasifikasikan penyimpangan tumbuh kembang anak menggunakan metode ELM. Data yang digunakan berjumlah 100 data dengan 38 parameter tentang gejala penyimpangan tumbuh kembang anak. Hasil dari pengujian menunjukkan bahwa rasio data latih dan data uji hanya memiliki pengaruh sedikit terhadap akurasi, perubahan jumlah *hidden neuron* berpengaruh pada waktu yang dibutuhkan untuk perhitungan ELM, dan fungsi aktivasi yang paling baik untuk data yang bernilai biner adalah fungsi aktivasi *sigmoid* biner. Setelah melakukan pengujian tersebut, maka dibuatlah *confusion matrix* dari nilai parameter yang paling optimal dari pengujian. Hasilnya, sistem mampu melakukan klasifikasi terbaik yaitu dengan nilai akurasi 76,67%.

Tabel 2.1 Kajian Pustaka

No.	Penulis	Objek	Metode	Hasil (*Output*)
1.	(Anggraeni & Indrarti, 2010)	Data laporan bulanan kasus balita gizi buruk pada Februari 2010 menggunakan 5 parameter input.	*Backpropagation*	Hasil: identifikasi status gizi balita Akurasi: 93,85% dengan data sampel berjumlah 130 data.

2.	(Irsyadi & Fathina, 2015)	Data hasil pemeriksaan gizi balita di Posyandu Lestari Asih Kartasutra, Sukoharjo menggunakan 3 parameter input.	*Backpropagation*	Hasil: identifikasi status gizi balita ke dalam 4 pilihan kelas. Akurasi: 100%.
3.	(Amalia, 2015)	Data penyakit diabetes dengan banyak input 9.	*Backpropagation* dan Algoritme Genetika	Hasil: prediksi penyakit diabetes ke dalam 2 pilihan kelas. Akurasi: 74,46% untuk metode *backpropagation* dan 77,10% untuk metode *backpropagation* dan Algoritme Genetika.
4.	(Fadilla, Adikara, & Perdana, 2018)	Data penyakit *chronic kidney disease* (CKD) dengan 24 parameter input.	ELM	Hasil: identifikasi penyakit CKD ke dalam 2 pilihan kelas. Akurasi: 96,7%.
5.	(Chandra, Santoso, & Adinugroho, 2018)	Data kualitas air sungai dengan 7 parameter input.	ELM dan Algoritme Genetika.	Hasil: identifikasi kualitas air sungai menjadi 3 kelas. Akurasi: 88,0002%.
6.	(Ariestyani, Adikara, & Perdana, 2018)	Objek: data penyimpangan tumbuh kembang pada balita dengan jumlah input sebanyak 38.	ELM	Hasil: identifikasi penyimpangan tumbuh kembang balita ke dalam 3 kelas. Akurasi: 76,67%.

Berdasarkan hasil kajian pustaka di atas, perbedaan penelitian ini dengan penelitian-penelitian sebelumnya yaitu terletak pada parameter penentu klasifikasi status gizi dan algoritme yang digunakan. Pada penelitian ini, parameter yang digunakan adalah jenis kelamin, umur, berat badan, dan panjang badan atau tinggi badan. Untuk pemilihan algoritme, berdasarkan hasil penelitian di atas maka dipilih ELM dan Algoritme Genetika sebagai solusi dalam penyelesaian permasalahan mengenai klasifikasi status gizi pada balita.

2.2 Klasifikasi

Klasifikasi merupakan metode yang biasa digunakan pada *data mining* dan *machine learning*. Klasifikasi termasuk metode pembelajaran yang terawasi (*supervised*), karena label kelas pada data latih (*training*) sudah diketahui. Terdapat banyak metode yang dapat digunakan untuk menyelesaikan masalah klasifikasi, diantaranya K-*Nearest Neighbor* (K-NN), Naive Bayes *Classification*, *backpropagation Neural Network*, *Learning Vector Quantization* (LVQ), dan *Extreme Learning Machine*.

2.3 Status Gizi

Status gizi merupakan suatu ekspresi untuk mengetahui tingkat keadaan kesehatan individu, yang umumnya ditentukan oleh keseimbangan antara asupan protein dan energi yang diperoleh dari makanan, yang kemudian dapat diukur secara antropometri. Di Indonesia, antropometri merupakan cara yang paling umum digunakan untuk melakukan penilaian status gizi karena sederhana, aman, dan mudah dilakukan.

2.3.1 Variabel Pengukuran Status Gizi

Dalam melakukan penilaian status gizi, antropometri disajikan dalam bentuk indeks yang dikaitkan dengan variabel lain. Variabel-variabel tersebut adalah sebagai berikut:

a. Jenis Kelamin

   Jenis kelamin turut berperan dalam penentuan status gizi, karena laki-laki dan perempuan memiliki lemak tubuh yang berbeda. Maka dari itu, pengukuran status gizi antara laki-laki dan perempuan berbeda.

b. Umur

   Umur memiliki peran yang penting dalam penentuan status gizi. Apabila penentuan umur salah, maka juga akan berpengaruh pada interpretasi status gizinya. Penentuan umur untuk penentuan status gizi yaitu 1 tahun adalah 12 bulan, 1 bulan adalah 30 hari. Jadi perhitungannya dalam bulan penuh, sisa umur dalam hari tidak diperhitungkan (Martiana, 2015). Misalkan seorang balita berumur 6 bulan 5 hari maka tetap ditulis 6 bulan.

c. Berat Badan

Berat badan adalah salah satu ukuran yang memberikan gambaran massa jaringan, termasuk jumlah protein, lemak, air, dan mineral pada tulang. Berat badan sangat sensitif terhadap perubahan yang mendadak baik karena penyakit infeksi maupun turunnya jumlah makanan yang dikonsumsi (Martiana, 2015).

d. Tinggi Badan

Tinggi badan merupakan parameter yang memberikan gambaran tentang fungsi pertumbuhan yang dapat dilihat dari keadaan kurus kering dan kecil pendek. Tinggi badan merupakan parameter yang penting bagi keadaan yang lalu dan keadaan sekarang, jika umur tidak diketahui dengan tepat. Parameter ini juga berkaitan erat dengan status sosial ekonomi, seperti kemiskinan.

2.3.2 Indikator Antropometri

Indikator antropometri yang kerap digunakan untuk menilai status gizi adalah berat badan menurut umur (BB/U), tinggi badan menurut umur (TB/U), berat badan menurut tinggi badan (BB/TB), dan lingkar lengan atas menurut umur (LLA/U) (Anggraeni & Indrarti, 2010). Namun pada penelitian ini, hanya menggunakan 3 indikator, yakni BB/U, TB/U, dan BB/TB.

1. Berat Badan Menurut Umur (BB/U)

   BB/U memberikan gambaran tentang berat badan balita yang dicapai pada umur tertentu. Dalam keadaan normal, berat badan akan berkembang mengikuti pertambahan umur. Sedangkan dalam keadaan abnormal, terdapat dua kemungkinan yaitu berat badan berkembang cepat atau lebih lambat dari keadaan normal. BB/U lebih menggambarkan status gizi seseorang saat ini.

2. Tinggi Badan Menurut Umur (TB/U)

   Indikator TB/U memberikan penjelasan tentang tinggi badan balita yang dicapai pada umur tertentu. Tidak seperti BB/U yang sensitif terhadap perubahan, TB/U relatif kurang sensitif terhadap masalah gizi dalam waktu pendek. Pengaruh defisiensi zat gizi terhadap tinggi badan akan nampak dalam waktu yang relatif lama, sehingga indeks ini menggambarkan status gizi seseorang pada masa lampau (Oktaviasari & Muniroh, 2009).

3. Berat Badan Menurut Tinggi Badan (BB/TB)

   BB/TB memberikan gambaran tentang berat badan balita yang dibandingkan dengan tinggi badan yang dicapai. Sama seperti BB/U, BB/TB juga dapat digunakan untuk menilai status gizi seseorang saat ini.

2.3.3 Kelas Status Gizi

Berdasarkan variabel-variabel di atas, maka nilai status gizi dapat dikelompokkan menjadi 8 kelas seperti yang ditunjukkan pada Tabel 2.2 (Sumber: Pakar).

Tabel 2.2 Kelas Status Gizi

Kelas	Kelas Status Gizi Menurut		
	BB/U	TB/U	BB/TB
1	Lebih	Normal	Gemuk
2	Normal	Normal	Gemuk
3	Normal	Pendek	Normal
4	Normal	Normal	Normal
5	Normal	Pendek	Gemuk
6	Kurang	Normal	Normal
7	Kurang	Normal	Kurus
8	Kurang	Pendek	Normal

2.4 Normalisasi Data

Sebelum dilakukan proses perhitungan dalam pengklasifikasian status gizi pada balita, perlu dilakukan proses normalisasi data. Normalisasi ini bertujuan untuk membuat standar yang sama untuk semua data yang digunakan dalam perhitungan, sehingga data berada pada *range* tertentu. Metode normalisasi yang digunakan dalam penelitian ini adalah *Min – Max Normalization*, yaitu metode yang mentransformasikan data ke dalam *range* 0 sampai 1. Proses normalisasi data dirumuskan seperti pada Persamaan (2.1).

$$X_i = \frac{data_{X_i} - data_{min}}{data_{max} - data_{min}} \quad \textbf{(2.1)}$$

Keterangan:

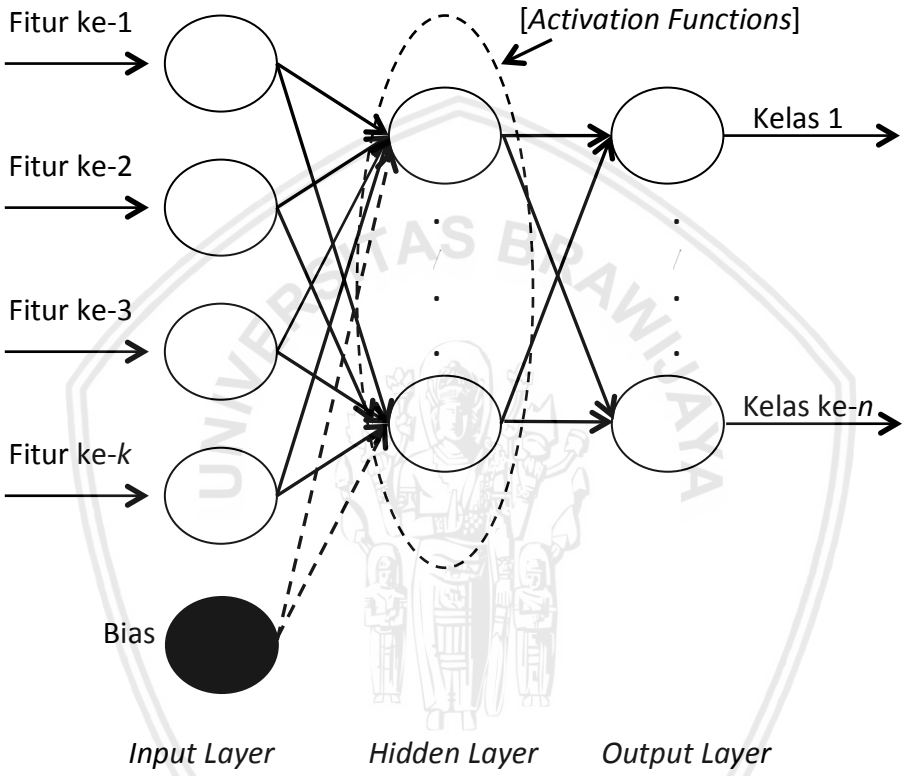
X_i = data ke-*i* yang telah ditransformasi

$data_{min}$ = nilai minimum dari data

$data_{max}$ = nilai maksimum dari data

2.5 *Extreme Learning Machine* (ELM)

ELM merupakan metode pembelajaran baru dari JST *feedforward* dengan satu *hidden neuron* atau biasa disebut dengan *single hidden neuron feedforward neural network* (SLFNs). Metode ini diciptakan untuk mengatasi kelemahan-kelemahan yang ada pada JST *feedforward*, terutama mengenai *learning speed* yang rendah (Huang, Zhu, & Siew, 2006). Hal tersebut dikarenakan penentuan parameter pada JST *feedforward* dilakukan secara iteratif menggunakan metode pembelajaran tersebut. Arsitektur ELM dapat dilihat pada Gambar 2.1 berikut:



Gambar 2.1 Arsitektur *Extreme Learning Machine*

Sumber: (Yaseen et al., 2018)

2.5.1 Fungsi Aktivasi *Sigmoid* Biner

Dalam JST, fungsi aktivasi digunakan untuk menentukan keluaran suatu *neuron*. Terdapat beberapa fungsi aktivasi yang digunakan dalam JST, seperti *sigmoid* biner, *hard limit*, sin, *sigmoid* bipolar, linear, radial basis, dan triangular basis. Pada penelitian ini digunakan fungsi aktivasi *sigmoid* biner karena JST yang dibangun membutuhkan nilai keluaran pada rentang 0 sampai 1. Rumus fungsi aktivasi *sigmoid* biner ditunjukkan pada Persamaan (2.2).

$$H = \frac{1}{(1 + exp^{-x.w^T})} \quad \textbf{(2.2)}$$

Keterangan:

H = *output hidden neuron* teraktivasi

x = data masukan

w^T = bobot masukan yang di-*transpose*

2.5.2 Proses *Training*

Proses *training* pada ELM dilakukan agar sistem melakukan pembelajaran atau pelatihan pada data latih yang diberikan. Langkah-langkah proses *training* metode ELM adalah sebagai berikut (Cholissodin & Riyandani, 2016).

1. Mencari bobot awal secara *random* untuk *w (weight)* sebagai bobot masukan.
2. Melakukan aktivasi matriks menggunakan fungsi aktivasi *sigmoid biner* seperti pada Persamaan (2.2).
3. Mencari nilai matriks *Moore-Penrose Pseudo Inverse* (H^+) menggunakan Persamaan (2.3).

$$H^+ = (H^T . H)^{-1} H^T \quad \textbf{(2.3)}$$

Keterangan:

H^T = matriks hasil aktivasi *output hidden neuron* yang di-*transpose.*

4. Menghitung nilai matriks *output weight* (β) menggunakan Persamaan (2.4).

$$\beta = H^+ T \quad \textbf{(2.4)}$$

Keterangan:

T = matriks target.

5. Menghitung hasil prediksi menggunakan Persamaan (2.5).

$$Y = H . \beta \quad \textbf{(2.5)}$$

Keterangan:

Y = hasil prediksi

2.5.3 Proses *Testing*

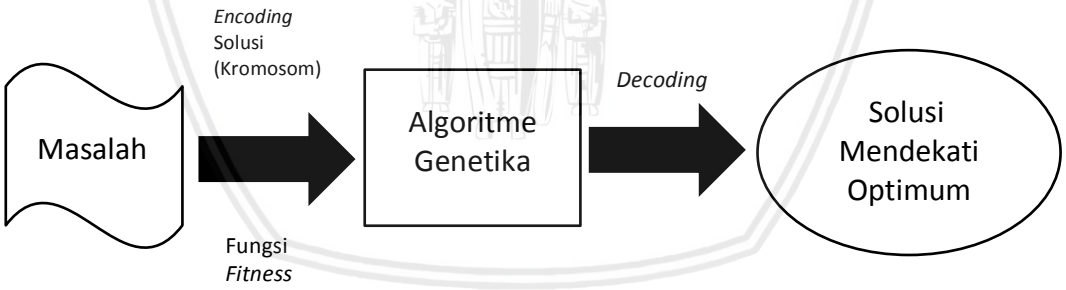
Setelah dilakukan proses *training*, kemudian langkah-langkah *testing* metode ELM adalah sebagai berikut (Cholissodin & Riyandani, 2016).

1. Mengambil nilai *w*, *b*, dan β yang dihasilkan dari proses *training*.
2. Melakukan aktivasi matriks menggunakan fungsi aktivasi *sigmoid biner* seperti pada Persamaan (2.2).
3. Menghitung hasil prediksi menggunakan Persamaan (2.5).

2.6 Algoritme Genetika

Algoritme Genetika sebagai cabang dari algoritme evolusi merupakan algoritme yang yang memanfaatkan mekanisme teori evolusi dan seleksi alam. Algoritme genetika memungkinkan untuk mengeksplorasi rentang potensi solusi yang jauh lebih besar untuk memecahkan suatu masalah daripada program konvensional (Holland, 2005). Dalam teori evolusi, dijelaskan bahwa terdapat sejumlah individu di suatu populasi. Para individu tersebut memiliki peran sebagai induk (*parent*) yang bereproduksi dan kemudian menghasilkan anak atau keturunan (*offspring*). Mereka (para individu dan *offspring*) kemudian berevolusi dan mengalami seleksi alam, dimana yang mampu beradaptasi dengan lingkungannya akan memiliki probabilitas yang lebih besar untuk dapat bertahan hidup. Dengan adanya seleksi alam tersebut maka individu yang lebih baik (yang mampu bertahan) memiliki probabilitas lebih tinggi untuk menghasilkan atau mendapatkan keturunan yang lebih baik pula. Sehingga, populasi dari setiap generasi akan lebih baik dari generasi sebelumnya. Algoritme ini tidak hanya dapat digunakan untuk menyelesaikan permasalahan optimasi pada rute, melainkan permasalahan apapun yang memerlukan solusi terbaik dan terefektif dari permasalahan pemilihan (Ramadonna, Sivia, & Ciksadan, 2017).

Untuk memecahkan suatu masalah menggunakan Algoritme Genetika, diawali dengan melakukan proses pemetaan pada solusi dari suatu masalah menjadi kromosom. Kromosom tersebut terdiri dari gen-gen yang menggambarkan variabel-variabel keputusan yang digunakan dalam solusi. Alur pencarian solusi pada Algoritme Genetika ditunjukkan pada Gambar 2.2 berikut:



Gambar 2.2 Diagram Alur Pencarian Solusi Algoritme Genetika

Sumber: (Wayan Firdaus Mahmudy, 2015)

2.6.1 Representasi dan Inisialisasi Kromosom

Inisialisasi kromosom bertujuan untuk membangkitkan populasi awal secara acak dan membentuk kromosom tertentu. Kromosom mewakili solusi dari permasalahan yang akan diselesaikan. Pada tahap ini juga dilakukan penentuan ukuran populasi (*popsize*). Terdapat dua metode dalam merepresentasikan kromosom, yaitu dengan pengkodean biner (*binary-coded genetic algorithm*) dan pengkodean *real* (*real-coded genetic algorithm* (RCGA)). Contoh representasi pada RCGA dapat dilihat pada Gambar 2.3.

0.32	0.23	0.93	0.944	0.22

Gambar 2.3 Contoh Representasi Kromosom RCGA

2.6.2 Reproduksi

Reproduksi bertujuan untuk menghasilkan keturunan (*offspring*) dari individu yang ada di populasi. Pada reproduksi, terdapat dua proses yang harus dilalui, yaitu *crossover* dan mutasi. Terdapat banyak metode yang telah dikembangkan untuk melakukan kedua proses tersebut, salah satunya adalah *Extended Intermediate Crossover* untuk *crossover* dan *Random Mutation* untuk mutasi. Sebelum melakukan kedua proses tersebut, harus ditentukan terlebih dahulu probabilitas *crossover* (*crossover rate* atau *cr*) dan probabilitas mutasinya (*mutation rate* atau *mr*). Kedua probabilitas tersebut berfungsi untuk menentukan rasio *offspring* yang dihasilkan, dengan cara mengalikan nilai *cr* atau *mr* dengan *popSize*.

1. *Crossover (Extended Intermediate Crossover)*

   Proses ini melibatkan dua buah *parent* dan nilai alpha (α) yang dipilih secara acak. Kedua *parent* tersebut dikawinkan secara silang dan menghasilkan kromosom baru, yang dihitung menggunakan Persamaan (2.6).

$$C_1 = P_1 + \alpha(P_2 - P_1)$$
$$C_2 = P_2 + \alpha(P_1 - P_2) \quad \textbf{(2.6)}$$

   Keterangan:
   C_1 = hasil *crossover* ke-1
   C_2 = hasil *crossover* ke-2
   P_1 = *parent* ke-1
   P_2 = *parent* ke-2
   α = nilai *alpha* yang dibangkitkan secara *random* dengan *range* [-0.25, 1.25]

   Gambar 2.4 merupakan contoh proses *extended intermediate crossover.*

P_1	0.32	0.23	0.93	0.44	0.22
P_2	0.20	0.11	0.31	0.29	0.89

α	1.23	0.12	-0.3	0.99	-0.1

C_1	0.17	0.22	1.12	0.29	0.15
C_2	0.35	0.12	0.12	0.44	0.96

Gambar 2.4 Contoh Proses *Extended Intermediate Crossover*

2. *Mutation (Random Mutation)*

   Proses ini melibatkan satu buah *parent*, nilai *random* (*r*), serta nilai maksimum dan minimum dari gen variabel x_i. Dengan adanya mutasi, maka keragaman populasi dapat terjaga (Wayan Firdaus Mahmudy, 2015). Untuk

mendapatkan hasil dari mutasi, dapat dihitung menggunakan Persamaan (2.7).

$$x'_i = x'_i + r(max_i - min_i) \quad \textbf{(2.7)}$$

Keterangan:
x'_i = *parent* yang terpilih
r = nilai *random* dengan *range* [-0.1, 0.1]
max_i = nilai maksimum dari gen variabel x_i
min_i = nilai minimum dari gen variabel x_i

Gambar 2.5 berikut merupakan contoh dari proses mutasi menggunakan *random mutation* dengan P_1 sebagai induk, gen yang terpilih nomor 1, dan r = -0.06.

P_1	**0.32**	0.23	0.93	0.44	0.22

C_3	**0.20**	0.23	0.93	1.93	2.22

Gambar 2.5 Contoh Proses *Random Mutation*

2.6.3 Evaluasi

Proses evaluasi digunakan untuk melakukan penilaian kepada setiap individu atau kromosom dalam populasi, apakah individu tersebut layak atau tidak untuk dipertahankan. Untuk mengukur suatu individu tersebut diperlukan sebuah nilai yang disebut *fitness*. Semakin besar nilai *fitness*, semakin besar pula peluang kromosom tersebut menjadi kandidat solusi. Pada penelitian ini, nilai *fitness* didapatkan dari hasil akurasi sistem. Sehingga, rumus *fitness* dibuat seperti pada Persamaan (2.8) (Chandra et al., 2018).

$$f(x) = Akurasi\ ELM = \frac{jumlah\ data\ benar}{total\ data} \quad \textbf{(2.8)}$$

2.6.4 Seleksi

Proses seleksi bertujuan untuk mendapatkan kromosom-kromosom yang berkualitas untuk dipertahankan pada generasi selanjutnya. Semakin tinggi nilai *fitness* suatu kromosom, maka semakin tinggi pula probabilitas kromosom tersebut terpilih. Proses seleksi dilakukan dengan menggunakan metode *Etilism Selection*, yaitu dengan mengurutkan seluruh individu dengan nilai *fitness* terbesar hingga terkecil, kemudian diambil beberapa individu sebanyak jumlah *popSize* dari populasi yang telah diurutkan, kemudian diproses pada generasi selanjutnya.

2.7 *Extreme Learning Machine* dan Algoritme Genetika

Seperti yang telah dijelaskan sebelumnya bahwa pada penelitian ini Algoritme Genetika digunakan untuk mengoptimasi bobot pada ELM yang

didapatkan secara *random*. Adapun langkah-langkah yang digunakan untuk melakukan perhitungan gabungan antara metode ELM dan Algoritme Genetika adalah sebagai berikut:

Langkah 1 Melakukan inisialisasi *popSize*, jumlah generasi, serta nilai *cr* dan *mr*.

Langkah 2 Melakukan inisialisasi kromosom awal.

Langkah 3 Melakukan proses reproduksi, yang terdiri dari proses *crossover* yang menggunakan *Extended Intermediate Crossover,* dan mutasi yang menggunakan *Random Mutation*.

Langkah 4 Melakukan proses evaluasi masing-masing kromosom dengan menggunakannya sebagai bobot awal metode ELM.

Langkah 5 Melakukan proses seleksi untuk mendapatkan individu dengan *fitness* tertinggi sejumlah *popSize*.

Langkah 6 Mengulang langkah di atas hingga generasi maksimum.

Langkah 7 Setelah mendapatkan kromosom terbaik, lakukan proses *training* pada ELM dengan langkah-langkah sebagai berikut:

a. Hitung matriks aktivasi data *training* menggunakan Persamaan (2.2).
b. Hitung matriks *Moore-Penrose Pseudo Inverse* menggunakan Persamaan (2.3).
c. Hitung bobot keluaran menggunakan Persamaan (2.4).

Langkah 8 Setelah melakukan proses *training*, dilakukan proses *testing* pada metode ELM dengan langkah-langkah sebagai berikut:

a. Hitung matriks aktivasi *output hidden neuron* dengan data *testing* menggunakan Persamaan (2.2).
b. Lakukan proses prediksi menggunakan Persamaan (2.5).

Langkah 9 Setelah mendapatkan hasil prediksi kelas, maka proses selesai.

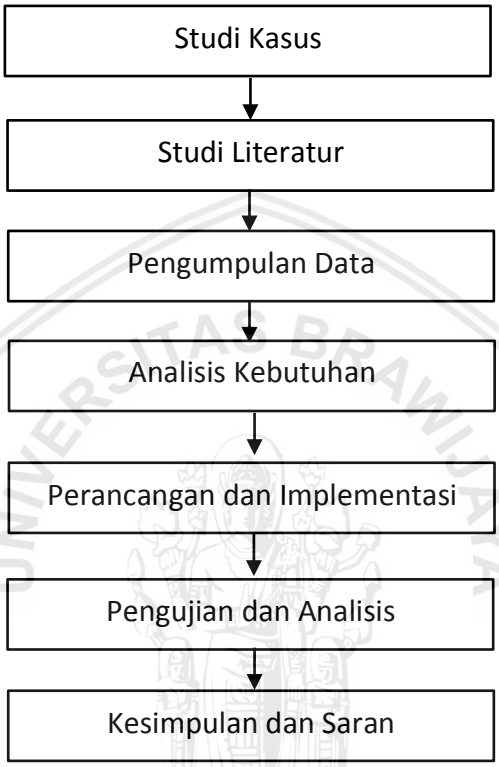
2.8 Nilai Evaluasi

Setelah semua tahapan dilalui, maka dilakukan pengujian untuk mengetahui kinerja dari metode yang digunakan. Pengujian dilakukan dengan cara sebagai berikut:

1. Pengujian konvergensi, digunakan untuk mengetahui pada iterasi berapa nilai *fitness* konvergen dan stagnan untuk menghasilkan akurasi terbaik.
2. Pengujian ukuran populasi, digunakan untuk menentukan ukuran populasi yang terbaik untuk menghasilkan akurasi terbaik.
3. Pengujian kombinasi nilai *cr* dan *mr*, digunakan untuk mengetahui kombinasi dari nilai *cr* dan *mr* yang terbaik untuk menghasilkan akurasi terbaik.
4. Pengujian jumlah *hidden neuron*, digunakan untuk mengetahui jumlah *hidden neuron* yang optimal untuk menghasilkan akurasi terbaik.

BAB 3 METODOLOGI

Bab metodologi menjelaskan tentang langkah-langkah yang digunakan dalam pembuatan sistem klasifikasi status gizi pada balita menggunakan *Extreme Learning Machine* dan Algoritme Genetika. Tahapan metodologi penelitian dalam penelitian ini ditunjukkan pada Gambar 3.1.



Gambar 3.1 Diagram Metodologi Penelitian

Berdasarkan Gambar 3.1, peneliti melakukan pencarian literatur yang dapat menunjang penyelesaian masalah penelitian, diantaranya teori tentang status gizi pada balita, metode *Extreme Learning Machine*, dan Algoritme Genetika. Setelah itu, melakukan pengumpulan data yang diperlukan pada penelitian kemudian menganalisis kebutuhan-kebutuhan yang diperlukan untuk melakukan perancangan dan implementasi. Pada perancangan dan implementasi, data pasien yang memeriksakan status gizinya akan diproses menggunakan metode *Extreme Learning Machine* dan Algoritme Genetika untuk menghasilkan sebuah keluaran status gizi yang sesuai dengan kondisi pasien tersebut. Dari hasil keluaran tersebut nantinya dilakukan pengujian untuk mengetahui kinerja dari sistem.

3.1 Tipe Penelitian

Tipe penelitian yang digunakan penulis adalah tipe non-implementatif analitik. Penelitian non-implementatif merupakan penelitian yang menitikberatkan pada investigasi terhadap keadaan atau fenomena tertentu. Teknik yang digunakan

peneliti adalah hasil dari studi kasus serta proses wawancara tentang objek penelitian yang digunakan. Sedangkan penelitian analitik adalah kegiatan penelitian yang memaparkan sebuah hasil analisis yang dihasilkan dari penjelasan hubungan antar elemen yang terdapat pada objek penelitian dengan kondisi tertentu yang diteliti. Secara garis besar, tipe non-implementatif analitik ini merupakan sebuah proses yang berkonsentrasi pada penggalian informasi dari suatu keadaan yang mampu menjawab pertanyaan-pertanyaan yang muncul pada awal penelitian.

3.2 Strategi Penelitian

Penelitian dimulai dari pengumpulan studi kasus seputar penentuan status gizi pada balita. Studi kasus ini didapatkan dari buku, internet, jurnal, maupun referensi-referensi lainnya, serta pakar. Hasil dari studi kasus tersebut digunakan sebagai permasalahan utama dalam penelitian ini, yang kemudian dilanjutkan pada proses pembelajaran studi literatur dalam pencarian algoritme untuk menyelesaikan permasalahan tersebut. Hasil dari studi pustaka ini selanjutkan diaplikasikan ke dalam sebuah sistem cerdas untuk menyelesaikan permasalahan.

3.3 Partisipan Penelitian

Partisipan penelitian yang terlibat dalam penelitian adalah ahli gizi yang terdapat di Puskesmas Pandanwangi Kota Malang, berperan sebagai penyedia data serta sumber informasi dalam bidang kesehatan terutama mengenai pengklasifikasian status gizi pada balita.

3.4 Lokasi Penelitian

Penelitian ini dilakukan di Puskesmas Pandanwangi Kota Malang selaku tempat penyedia data, dan di lingkungan Fakultas Ilmu Komputer Universitas Brawijaya selaku tempat pelaksanaan penelitian.

3.5 Teknik Pengumpulan Data

Pada tahap pengumpulan data, peneliti menggunakan data primer, yaitu data yang diperoleh secara langsung dari instansi terkait melalui tahap wawancara dengan ahli gizi, serta mengkaji data tersebut melalui studi literatur.

3.6 Data Penelitian

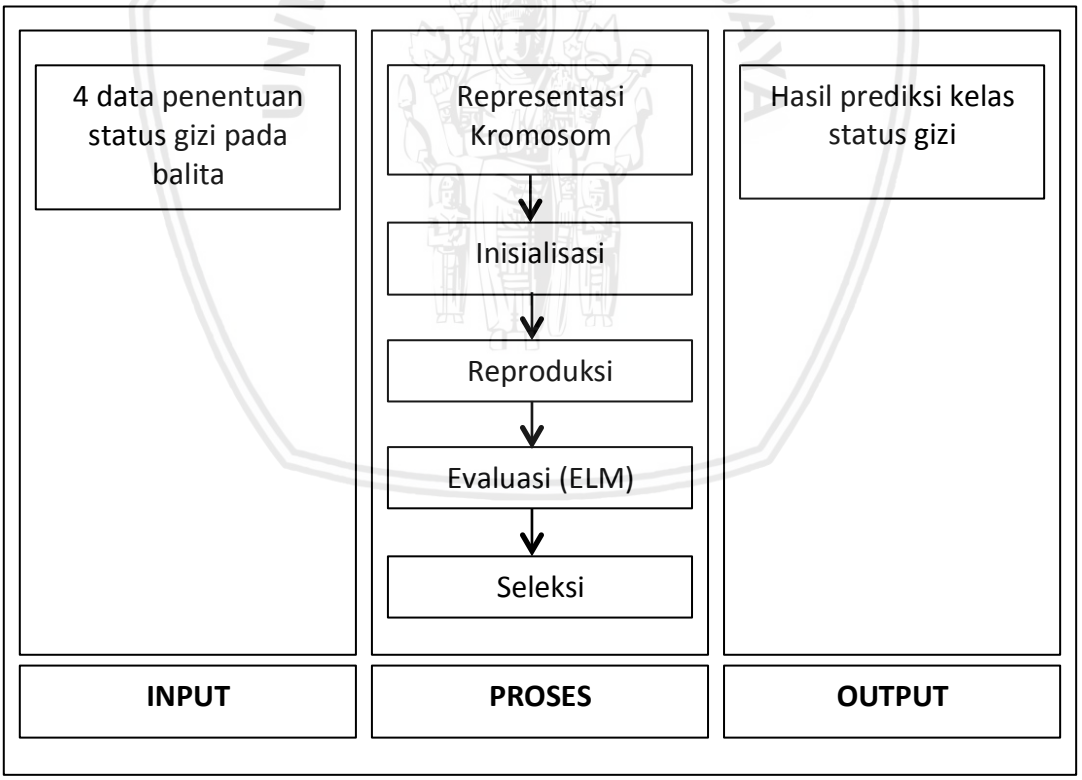
Data yang digunakan dalam penelitian ini adalah data rekam medis Puskesmas Pandanwangi Kota Malang pada Agustus 2018. Total data yang diperoleh sejumlah 118 data dengan 4 parameter masukan, yaitu jenis kelamin, umur (bulan), berat badan (kg), dan tinggi badan (cm) atau panjang badan (cm). Dari 118 data tersebut, dibagi menjadi 8 kelas dengan rincian jumlah data sebagai berikut:

1. Kelas 1 = 5 data.

2. Kelas 2 = 4 data.
3. Kelas 3 = 14 data.
4. Kelas 4 = 78 data.
5. Kelas 5 = 3 data.
6. Kelas 6 = 6 data.
7. Kelas 7 = 4 data.
8. Kelas 8 = 8 data.

3.7 Teknik Analisis Data

Teknik analisis data dilakukan melalui proses pengujian dengan data yang digunakan. Hasil perhitungan data kemudian dilakukan pengujian dengan 4 skenario pengujian, yaitu pengujian konvergensi, ukuran populasi, kombinasi nilai *cr* dan *mr,* serta jumlah *hidden neuron*. Berikut adalah model perancangan sistem yang menjelaskan tentang cara kerja sistem secara garis besar, mulai dari *input* yang dimasukkan hingga mendapatkan hasil. Diagram perancangan sistem dapat dilihat pada Gambar 3.2.



Gambar 3.2 Diagram Perancangan

Penjelasan tentang diagram perancangan yang terdapat pada Gambar 3.2.

Gambar 3.2 adalah sebagai berikut:

Input : 4 data penentuan status gizi, yakni jenis kelamin, umur, berat

badan, dan tinggi badan.

Proses : Sistem akan melakukan perhitungan bobot menggunakan Algoritme Genetika melalui proses representasi kromosom, inisialisasi, serta reproduksi. Kemudian hasil dari proses tersebut masing-masing akan dievaluasi menggunakan metode ELM untuk mendapatkan nilai *fitness*-nya. Setelah didapatkan *fitness*, bobot akan masuk ke tahap seleksi. Proses tersebut berulang hingga iterasi yang telah ditentukan.

Output : Keluaran sistem berupa prediksi kelas status gizi berdasarkan perhitungan yang telah dilakukan.

3.8 Implementasi Algoritme

Tahapan awal dalam proses implementasi algoritme adalah dengan melakukan perhitungan manual dengan tujuan untuk memudahkan peneliti dalam memahami setiap langkah perhitungan dalam metode ELM dan Algoritme Genetika. Tahap selanjutnya adalah membuat *flowchart* untuk menentukan alur perhitungan metode. Selanjutnya melakukan implementasi sistem dengan menggunakan bahasa pemrograman Java.

3.9 Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi, dan pengujian metode. Kesimpulan diambil berdasarkan hasil pengujian dan analisis metode yang diterapkan dengan cara memastikan bahwa sistem berjalan sesuai dengan yang diharapkan dan juga membandingkan tingkat keakurasian tentang hasil dari sistem dengan analisis yang dilakukan oleh ahlinya. Kemudian saran diungkapkan ketika pengujian telah selesai dilakukan dengan tujuan untuk memperbaiki kesalahan-kesalahan yang terjadi selama pengujian. Dan diharapkan tidak terjadi pada penelitian berikutnya.

BAB 4 PERANCANGAN

Bab perancangan menjelaskan tentang seluruh proses perancangan penelitian, dimulai dari formulasi permasalahan, tahapan metode *Extreme Learning Machine* dan Algoritme Genetika, perhitungan manual dari algoritme yang digunakan, serta perancangan pengujian.

4.1 Formulasi Permasalahan

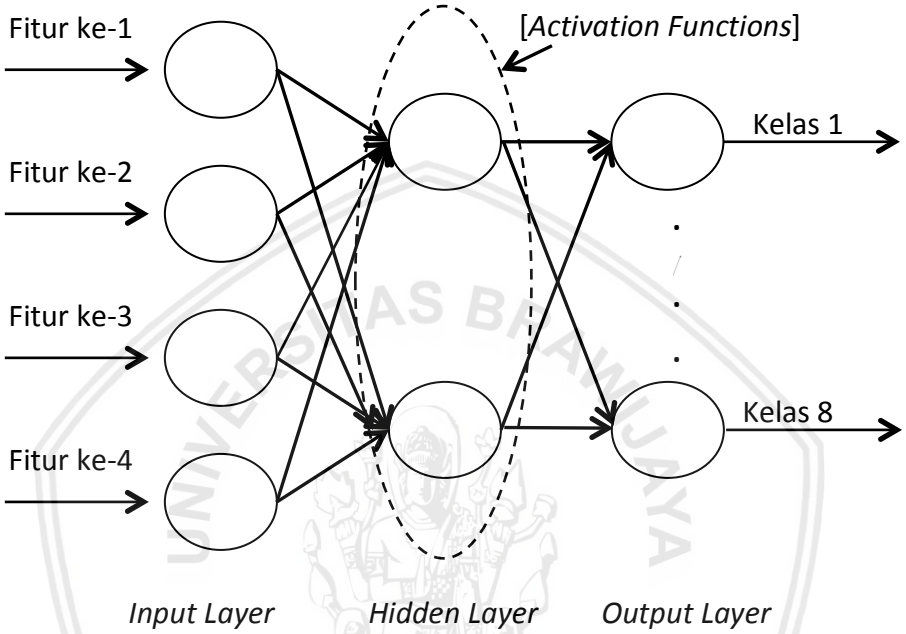
Permasalahan yang diangkat dalam penelitian ini yaitu penentuan status gizi pada balita. Guna mendapatkan penentuan status gizi yang cepat dan akurat, maka solusi yang digunakan yaitu pembuatan sebuah sistem cerdas yang mampu mengklasifikasikan status gizi pada balita. Sistem cerdas yang dibuat mengaplikasikan metode *Extreme Learning Machine* dan Algoritme Genetika. Tahap pertama yang dilakukan adalah melakukan perhitungan menggunakan Algoritme Genetika untuk mendapatkan bobot yang paling optimal. Langkah-langkah yang dilakukan adalah inisialisasi kromosom yang mencakup inisialisasi jumlah *popSize*, nilai *cr*, nilai *mr*, jumlah iterasi, proses reproduksi, evaluasi, dan seleksi. Individu yang terpilih melalui proses seleksi pada Algoritme Genetika kemudian digunakan sebagai bobot awal metode *Extreme Learning Machine* dalam melakukan klasifikasi. Adapun sampel data status gizi pada balita (diambil masing-masing 4 data dari kelas 1, 2, dan 3) berjumlah 12 data yang ditunjukkan pada Tabel 4.1 (Selengkapnya dapat dilihat pada lampiran).

Tabel 4.1 Sampel Data Status Gizi pada Balita

No.	Jenis Kelamin	Umur (bulan)	Berat Badan (kg)	Tinggi Badan (cm)	Kelas
1.	1	34	20.5	96	1
2.	1	55	29	110	1
3.	0	36	22	97	1
4.	1	59	31.5	112	1
5.	1	41	17	95	2
6.	1	56	2.2	107	2
7.	1	20	14	84	2
8.	0	56	19	102	2
9.	1	23	10.2	79	3
10.	0	23	9	78	3
11.	0	17	10	73	3
12.	0	33	12	82	3

4.2 Desain Arsitektur Sistem

Pada desain arsitektur metode *Extreme Learning Machine,* terdapat tiga *layer* (lapisan) yang memiliki fungsi berbeda. Untuk melakukan sebuah klasifikasi status gizi menggunakan metode *Extreme Learning Machine,* dibutuhkan tiga lapisan*,* yaitu *input layer*, *hidden layer*, dan *output layer*. Desain arsitektur sistem yang dibangun ditunjukkan pada Gambar 4.1.



Gambar 4.1 Desain Arsitektur ELM

Berikut merupakan penjelasan mengenai arsitektur metode ELM yang dibangun dalam melakukan klasifikasi status gizi pada Gambar 4.1.

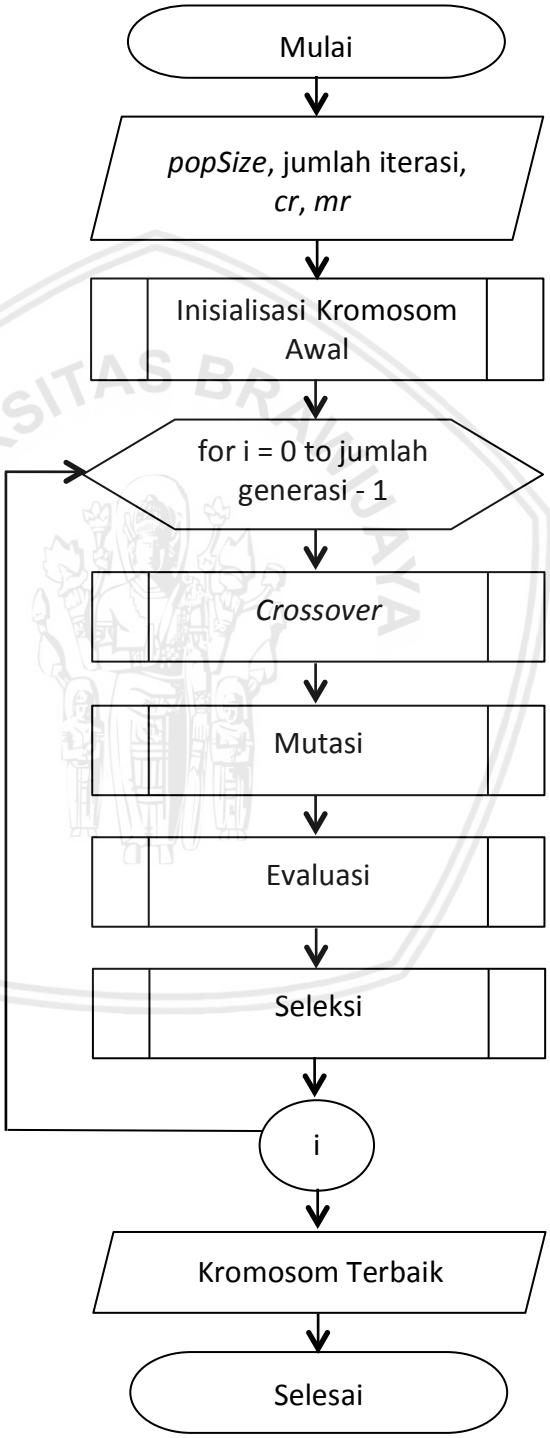
Input layer : Lapisan ini berisi masukan berupa data penentu klasifikasi status gizi. Terdiri dari 4 *neuron* karena melambangkan 4 variabel penentu status gizi. *Neuron* pertama melambangkan jenis kelamin, *neuron* kedua melambangkan umur, *neuron* ketiga melambangkan berat badan, dan *neuron* keempat melambangkan tinggi badan.

Hidden layer : Lapisan ini berfungsi untuk mengolah data masukan dari *input layer* dan menghasilkan *output* yang telah diolah menggunakan fungsi aktivasi. Jumlah *layer* yang digunakan pada *hidden layer* adalah 1, dengan *neuron* sebanyak 2.

Output layer : Lapisan ini merupakan lapisan yang memberikan hasil perhitungan prediksi kelas.

4.3 Siklus Algoritme Genetika

Siklus Algoritme Genetika berisi tentang proses yang dilalui untuk mendapatkan bobot yang optimal untuk perhitungan status gizi pada balita menggunakan metode ELM. Secara garis besar, aliran proses Algoritme Genetika pada penelitian ini dapat dilihat pada Gambar 4.2.

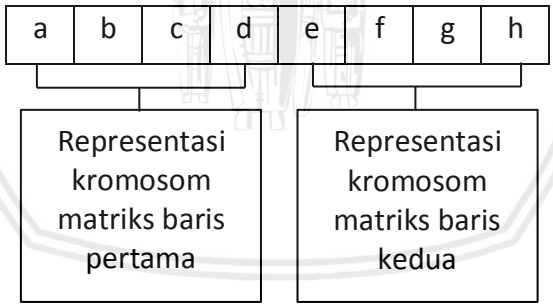


Gambar 4.2 *Flowchart* Algoritme Genetika

Gambar 4.2 menjelaskan tentang proses Algoritme Genetika secara keseluruhan. Algoritme genetika adalah pendekatan yang mampu melakukan optimasi variabel yang memiliki pengaruh terhadap suatu metode. Pada penelitian ini variabel yang akan dioptimasi adalah bobot awal dari metode ELM. Seperti yang tertera pada gambar bahwa proses Algoritme Genetika yaitu melakukan *input popSize*, jumlah generasi, *cr*, dan *mr*, kemudian melakukan inisialisasi kromosom awal. Setelah kromosom awal terinisialisasi, maka dilakukan proses *crossover*, mutasi, evaluasi, dan seleksi yang dilakukan secara iteratif. Setelah mencapai jumlah generasi tertentu, maka didapatkan kromosom terbaik.

4.3.1 Proses Representasi Kromosom

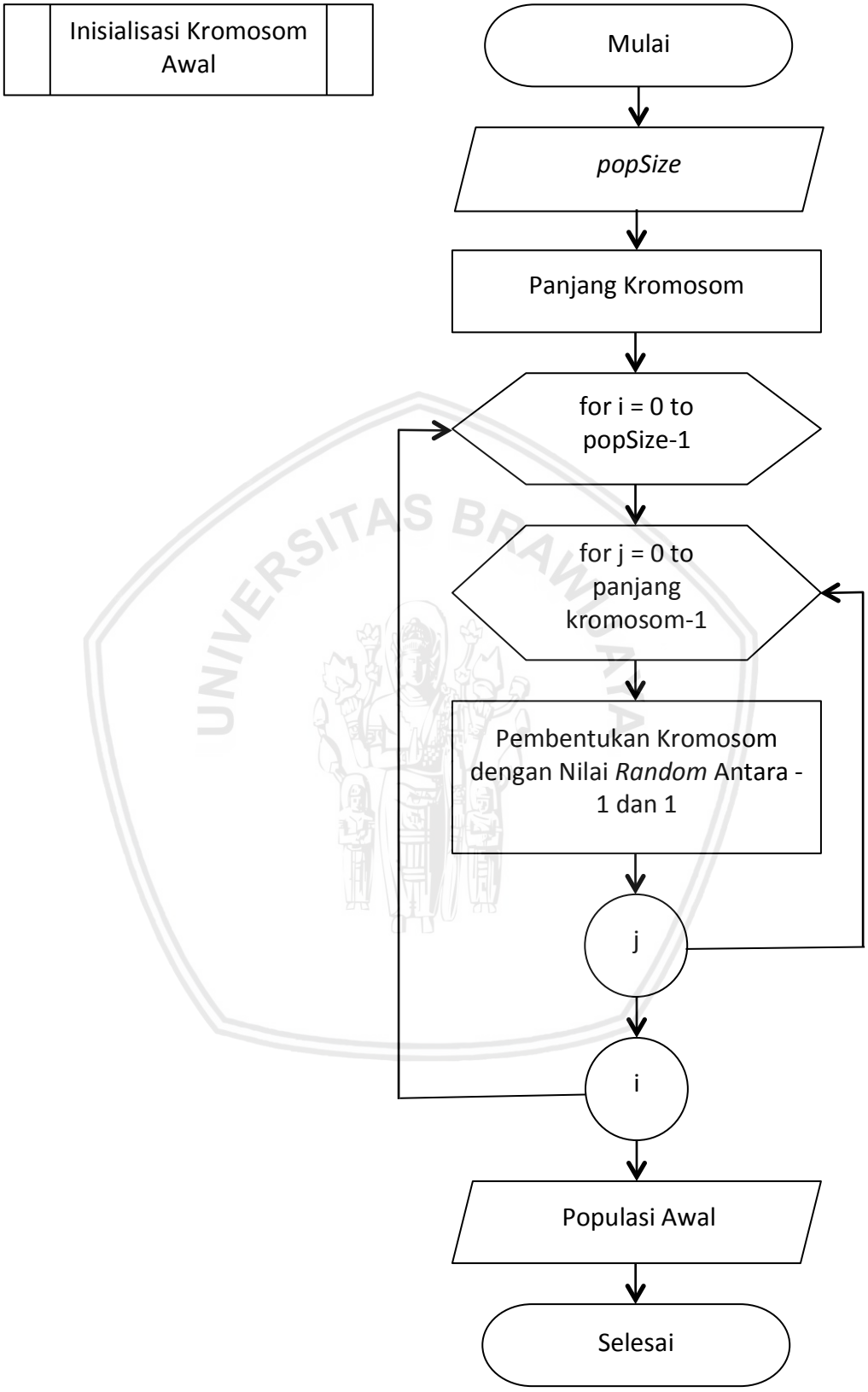
Representasi kromosom yang digunakan pada penelitian ini adalah representasi kromosom Algoritme Genetika dengan *Real-Coded Genetic Algorithm* (RCGA). Nilai dari kromosom dibangkitkan secara *random* dengan *range* -1 sampai dengan 1. Bobot awal pada ELM direpresentasikan dalam bentuk matriks dengan ukuran *j* x *k*, dimana *j* adalah jumlah *hidden neuron* dan *k* adalah jumlah *input layer*. Sebagai contoh, *hidden neuron* yang digunakan sebanyak 2, dan *input layer* yang digunakan sebanyak 4, yang merupakan representasi dari 4 parameter penentu status gizi pada balita. Dari ukuran tersebut, maka panjang kromosom yang direpresentasikan adalah sebanyak 2 x 4 yaitu 8. Sehingga matriks berordo 2 x 4 yang digunakan sebagai bobot awal adalah $\begin{bmatrix} a & b & c & d \\ e & f & g & h \end{bmatrix}$. Dari matriks tersebut, maka representasi kromosomnya ditunjukkan pada Gambar 4.3.



Gambar 4.3 Representasi Kromosom Menggunakan RCGA

4.3.2 Proses Inisialisasi Kromosom

Setelah kromosom terepresentasi, maka proses selanjutnya adalah inisialisasi kromosom. Proses inisialisasi kromosom dijelaskan pada Gambar 4.4.



Gambar 4.4 *Flowchart* Inisialisasi Kromosom

Gambar 4.4 merupakan alur proses inisialisasi kromosom pada Algoritme Genetika yang terdiri dari:

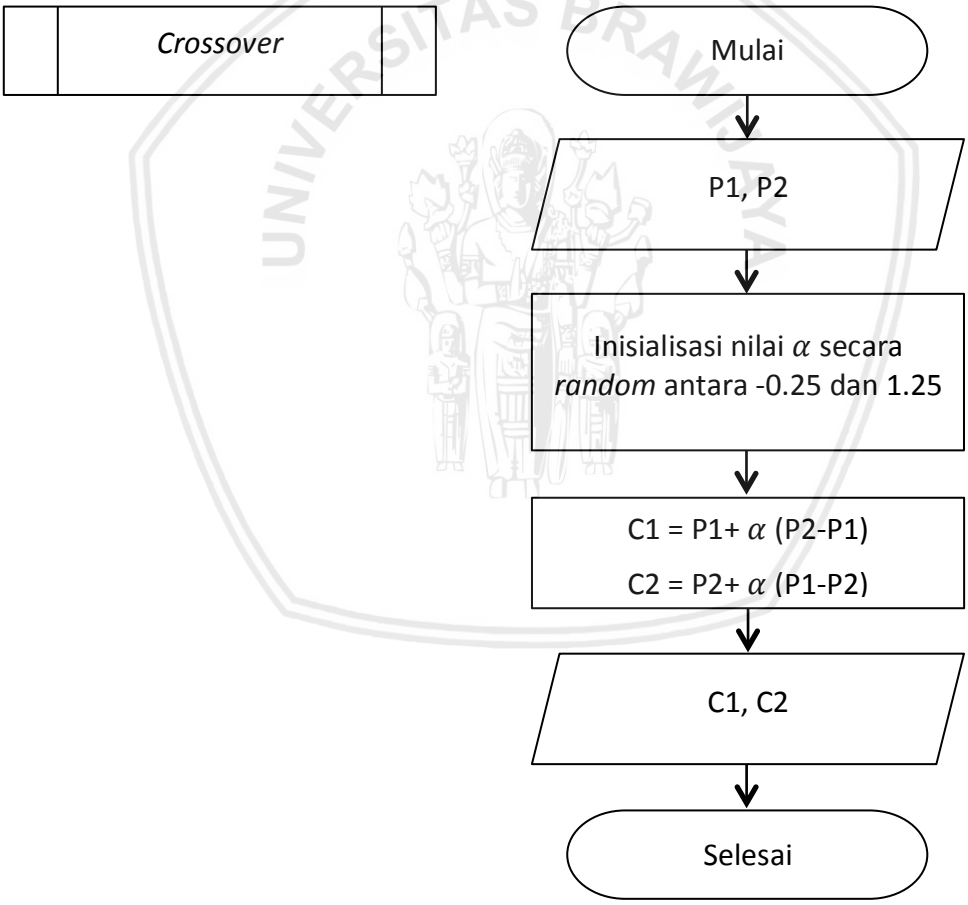
1. Menentukan jumlah populasi (*popSize*).
2. Menginisialisasi panjang kromosom, yang pada penelitian ini sebanyak 16 gen.
3. Membangkitkan nilai pada kromosom secara *random* dengan *range* [-1, 1].
4. Jika seluruh kromosom sudah memiliki nilai, maka terbentuklah populasi awal.

4.3.3 Proses Reproduksi

Terdapat 2 bagian dalam proses reproduksi, yaitu *crossover* dan mutasi yang siklusnya akan dijelaskan sebagai berikut:

4.3.3.1 Proses Crossover (Extended Intermediate Crossover)

Proses *crossover* menggunakan *extended intermediate crossover* dijelaskan pada Gambar 4.5.



Gambar 4.5 *Flowchart Crossover*

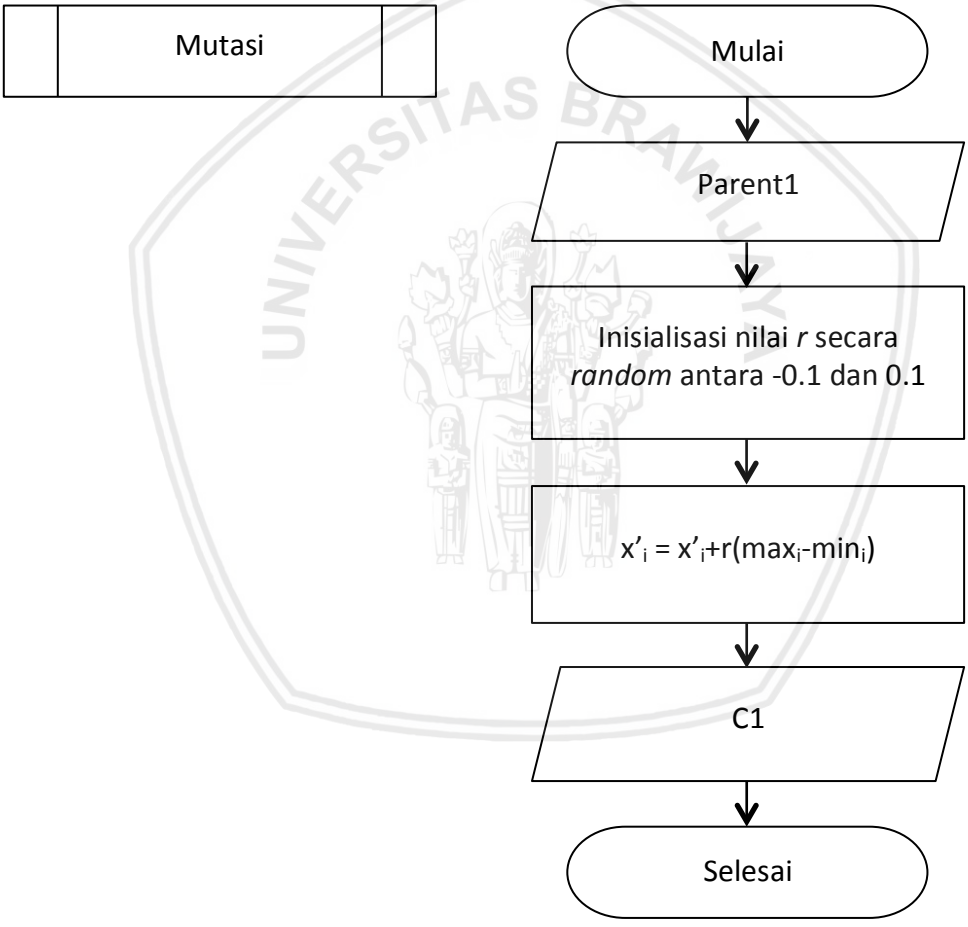
Berdasarkan Gambar 4.5, alur proses *crossover* dapat dijelaskan sebagai berikut:

1. Memilih dua buah *parent* yang dipilih dari populasi awal untuk dilibatkan dalam proses *crossover*.

2. Membangkitkan nilai *alpha* (α) secara *random* dengan *range* [-0.25, 1.25] sebanyak panjang satu kromosom.
3. Melakukan pembentukan *offspring* yang jumlahnya telah ditentukan, yaitu sejumlah *cr* x *popSize*. Perhitungan *crossover* ini dilakukan menggunakan *Extended Intermediate Crossover* yaitu dengan menjumlahkan masing-masing gen yang terdapat pada dua *parent* yang terpilih dengan perkalian antara nilai α dan selisih antara masing-masing gen pada *parent* 1 dan *parent* 2.
4. Setelah dilakukan perhitungan maka terbentuklah *offspring*.

4.3.3.2 Proses Mutasi (Random Mutation)

Setelah melakukan proses *crossover*, langkah selanjutnya yaitu mutasi yang dijelaskan pada Gambar 4.6.



Gambar 4.6 *Flowchart* Mutasi

Berdasarkan Gambar 4.6, langkah-langkah mutasi adalah sebagai berikut:

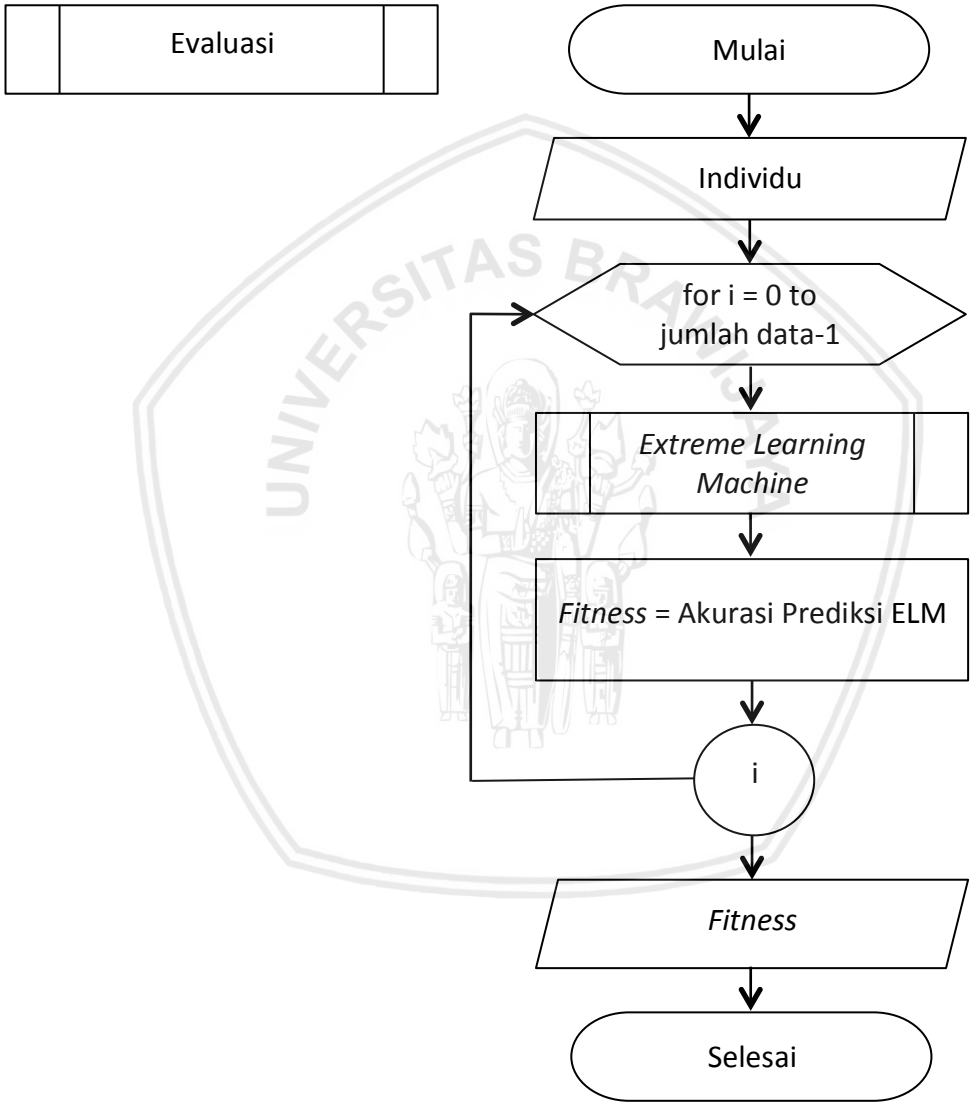
1. Memilih satu buah *parent* untuk digunakan dalam proses mutasi.
2. Membangkitkan nilai *r* secara *random* pada range [-0.1, 0.1].
3. Melakukan pembentukan *offsprin*g yang jumlahnya telah ditentukan, yaitu sejumlah *mr* x *popSize*. Proses mutasi ini menggunakan metode *Random*

Mutation, yaitu dengan menambahkan gen terpilih dengan hasil perkalian antara nilai *r* dan nilai maksimum dan minimum dari gen yang terpilih.

4. Proses perhitungan tersebut akan menghasilkan *offspring* baru.

4.3.4 Proses Evaluasi

Proses evaluasi akan menghasilkan nilai *fitness* yang digunakan untuk mengetahui apakah individu tersebut dapat terpilih ke generasi berikutnya atau tidak. Proses evaluasi lebih lanjut akan dijelaskan pada Gambar 4.7.



Gambar 4.7 *Flowchart* Evaluasi

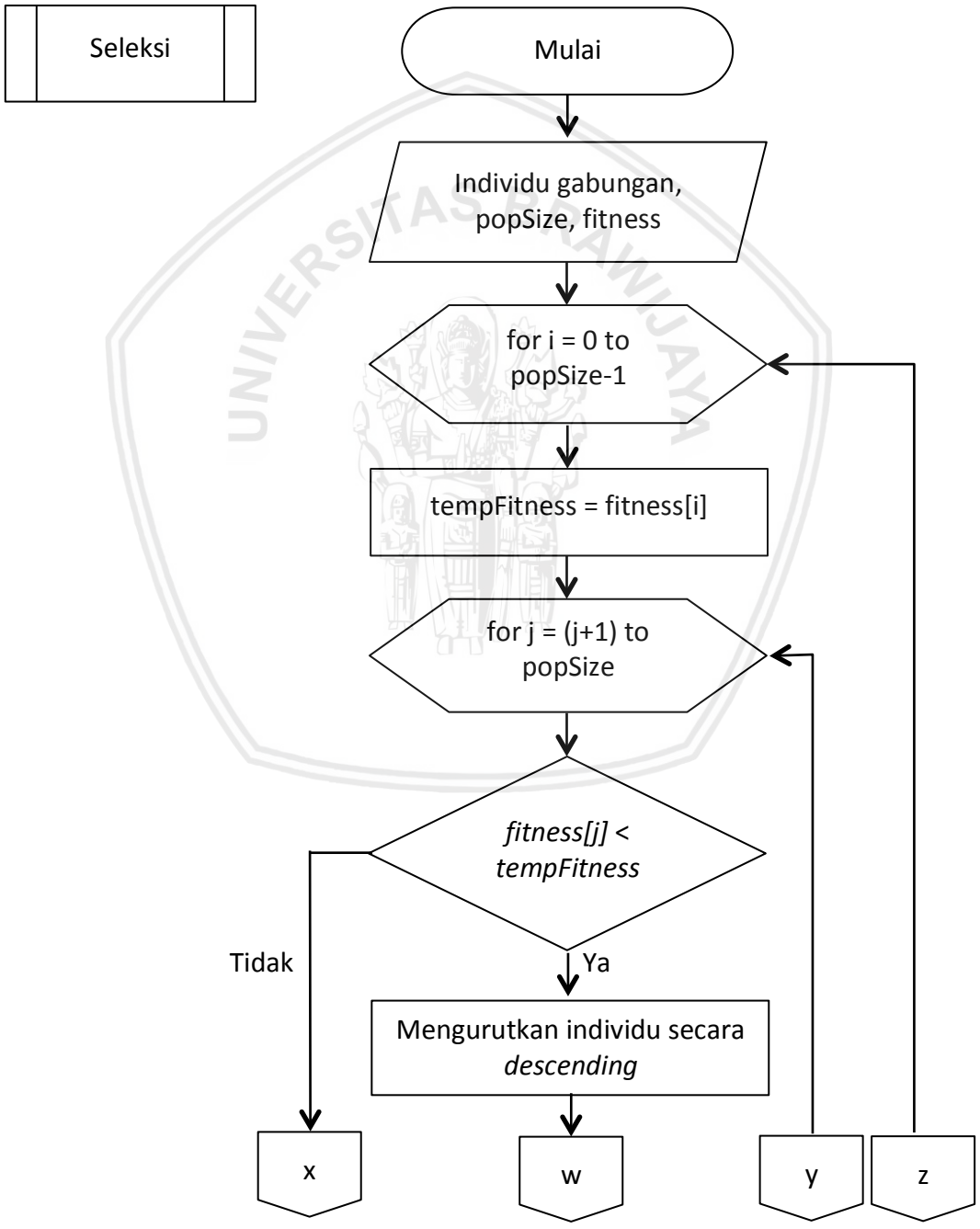
Berdasarkan Gambar 4.7, tahapan proses evaluasi dapat dijabarkan sebagai berikut:

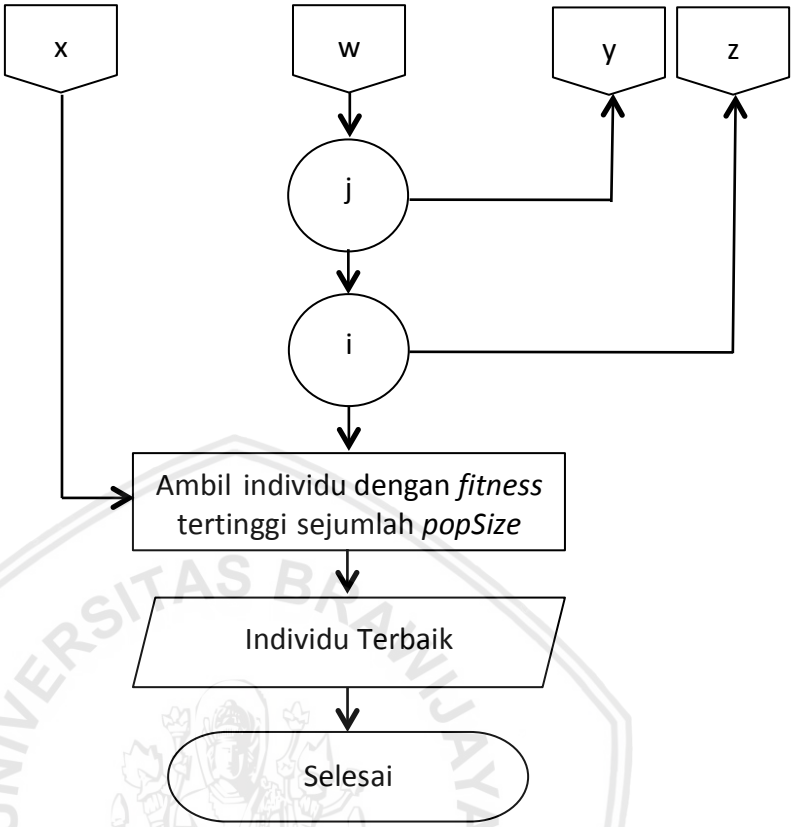
1. Inisialisasi individu (*parent* dan *offspring*) yang akan dicari nilai *fitness*-nya.
2. Memasukkan satu persatu individu ke dalam proses ELM sebagai bobot awal dan melakukan perhitungan sesuai metode ELM untuk melakukan klasifikasi status gizi pada balita.

3. Setelah dilakukan klasifikasi, hitung akurasinya dengan membandingkan jumlah kelas yang benar dengan total data. Hasil akurasi tersebut akan menjadi nilai *fitness* masing-masing individu.
4. Nilai *fitness* dari masing-masing individu telah didapatkan.

4.3.5 Proses Seleksi

Setelah didapatkan nilai *fitness*, langkah selanjutnya yaitu melakukan seleksi individu yang dapat bertahan ke generasi selanjutnya. Proses seleksi selengkapnya ditunjukkan pada Gambar 4.8.





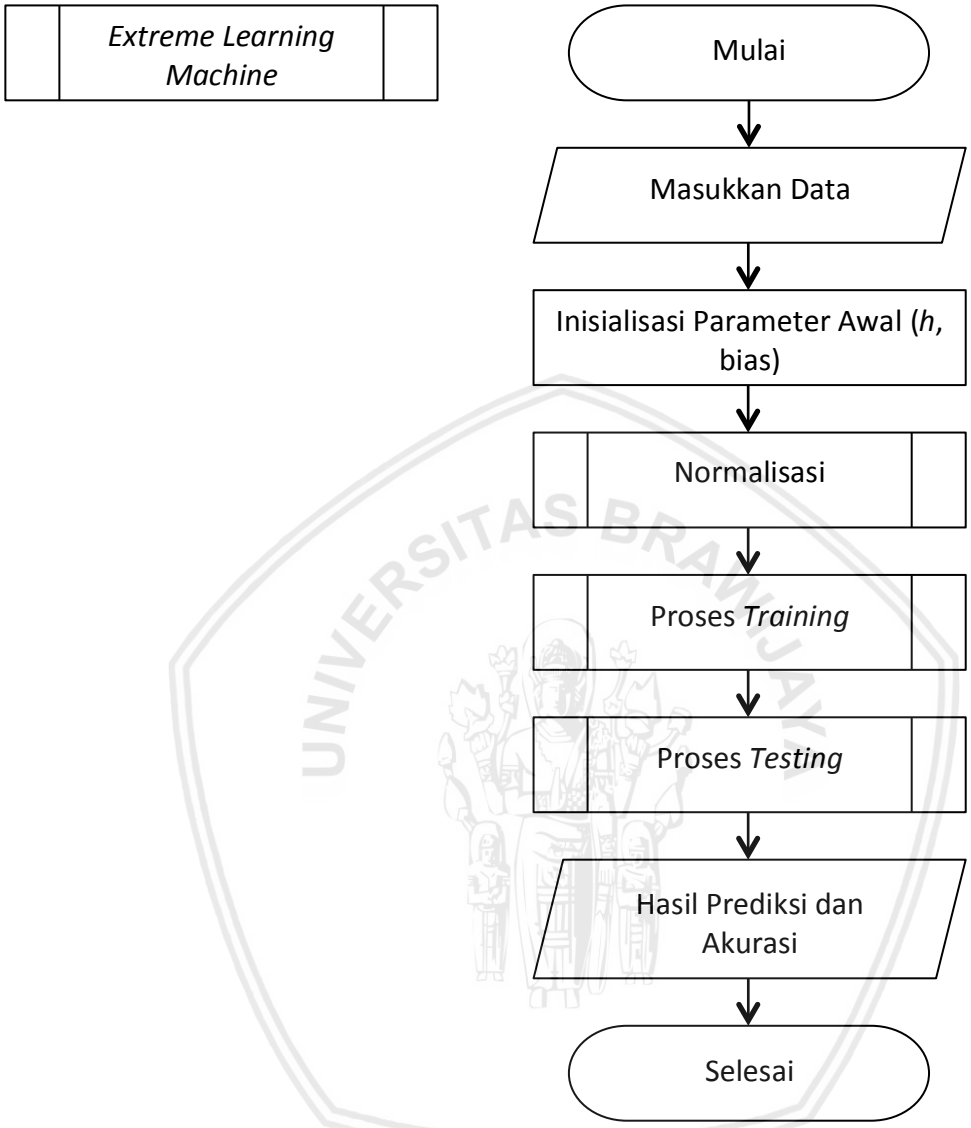
Gambar 4.8 *Flowchart* Seleksi

Penjelasan tentang proses seleksi seperti yang ditunjukkan pada Gambar 4.8 adalah sebagai berikut:

1. Masukkan individu gabungan (*parent* dan *offspring*), *popSize*, dan nilai *fitness*.
2. Melakukan pengurutan *(sorting)* individu berdasarkan nilai *fitness* secara *descending*, yaitu dari yang terbesar hingga yang terendah.
3. Ambil individu dengan *fitness* tertinggi sejumlah *popSize*.

4.4 Siklus Metode *Extreme Learning Machine* (ELM)

Siklus metode ELM berisi tentang proses yang digunakan untuk mengklasifikasikan status gizi pada balita setelah didapatkan bobot yang optimal dari perhitungan menggunakan Algoritme Genetika. Untuk aliran proses metode ELM yang digunakan pada penelitian ini dapat dilihat pada Gambar 4.9.

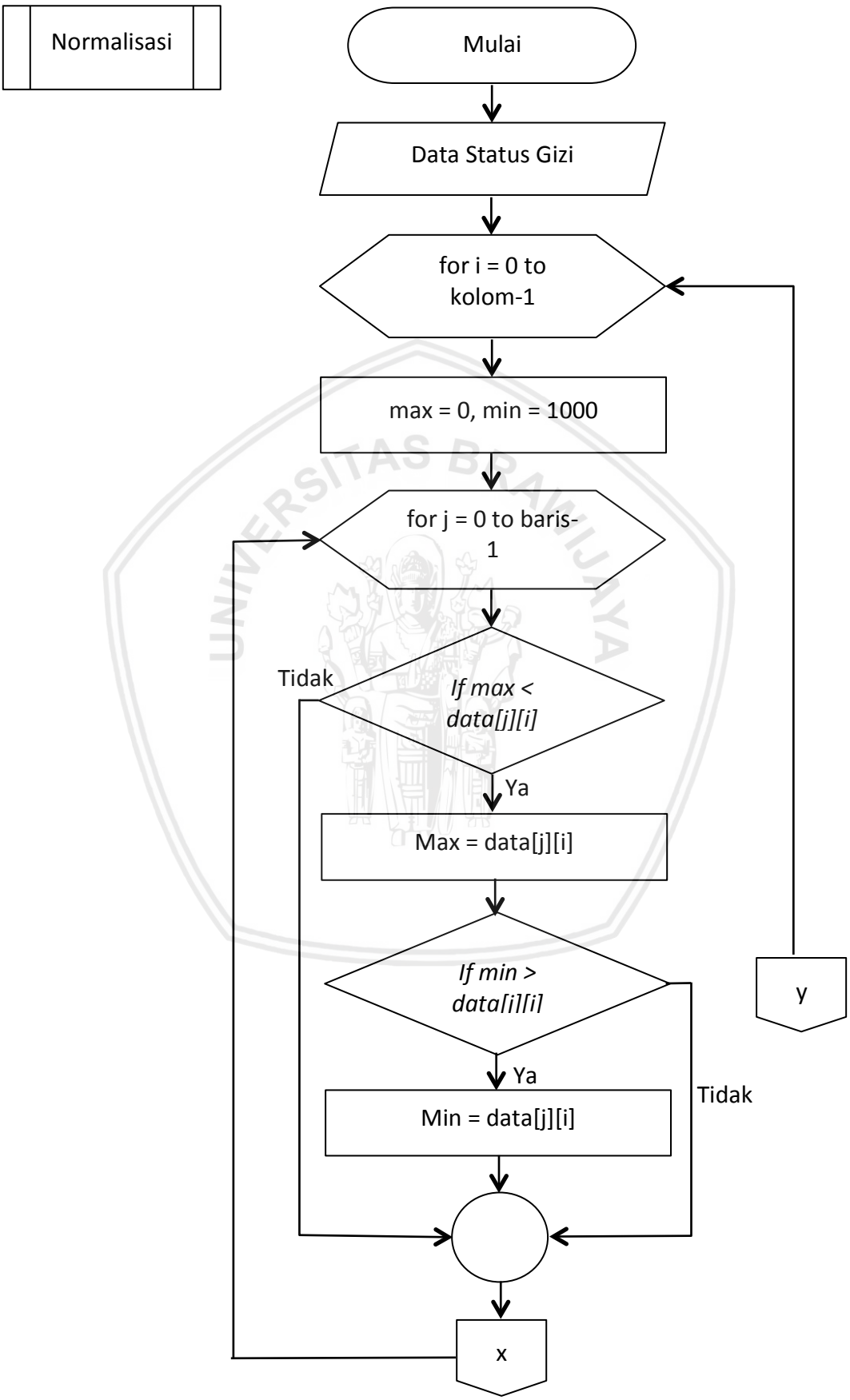


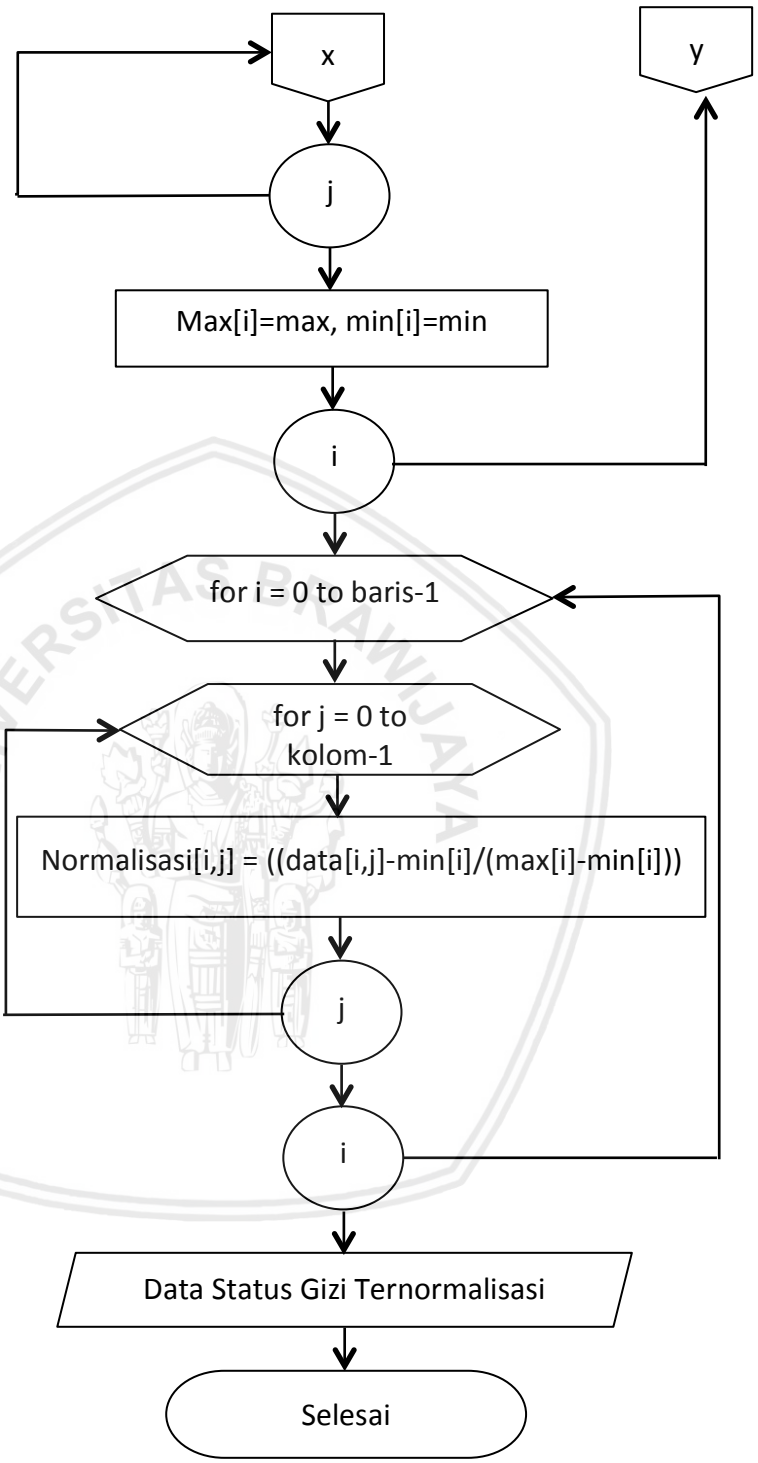
Gambar 4.9 *Flowchart* Metode *Extreme Learning Machine*

Gambar 4.9 menjelaskan tentang proses yang dilalui metode ELM dalam penyelesaian masalah. Pertama yaitu melakukan normalisasi data, baik data *training* maupun data *testing*. Kemudian melakukan proses *training* untuk melakukan pembelajaran kepada sistem, setelah itu proses *testing* untuk menguji data dan mendapatkan hasil prediksi kelas.

4.4.1 Proses Normalisasi Data

Normalisasi merupakan proses mengubah nilai dari suatu data dengan tujuan untuk mengurangi perbedaan rentang antar data pada setiap variabel. Metode normalisasi yang digunakan dalam penelitian ini adalah metode *Min-Max Normalization*. Proses normalisasi dapat dilihat pada Gambar 4.10.


Normalisasi
Mulai
Data Status Gizi
for i = 0 to kolom-1
max = 0, min = 1000
for j = 0 to baris-1
If max < data[j][i]
Tidak
Ya
Max = data[j][i]
If min > data[i][i]
Ya
Tidak
Min = data[j][i]
x
y



Gambar 4.10 *Flowchart* Normalisasi

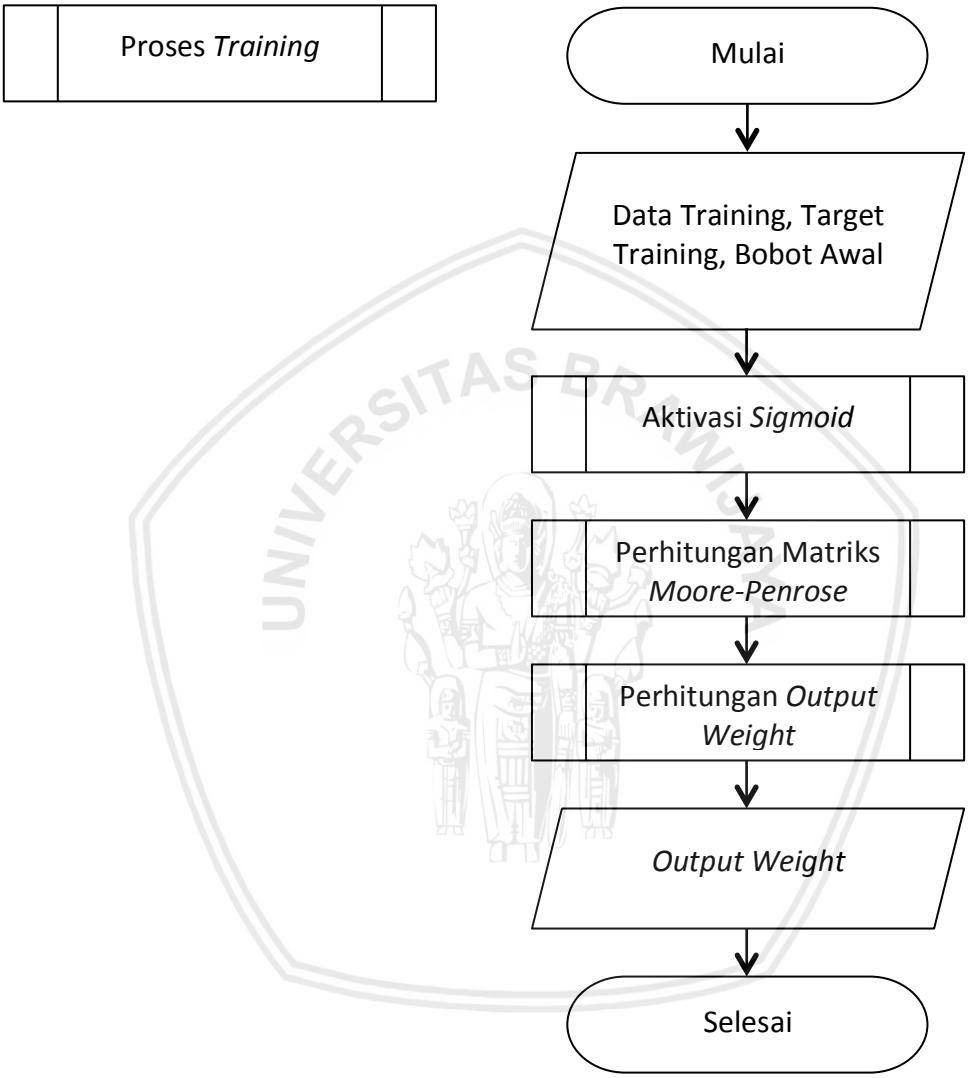
Berdasarkan Gambar 4.10, proses normalisasi memiliki langkah sebagai berikut:

1. Masukkan data status gizi pada balita.
2. Mencari nilai minimum dan maksimum dari tiap parameter.
3. Melakukan perhitungan normalisasi.

4. Mendapatkan hasil data status gizi ternormalisasi.

4.4.2 Proses *Training*

Proses *training* terdiri dari perhitungan aktivasi matriks, serta pencarian nilai matriks *Moore-Penrose*. Keseluruhan proses *training* dapat dilihat pada Gambar 4.11.



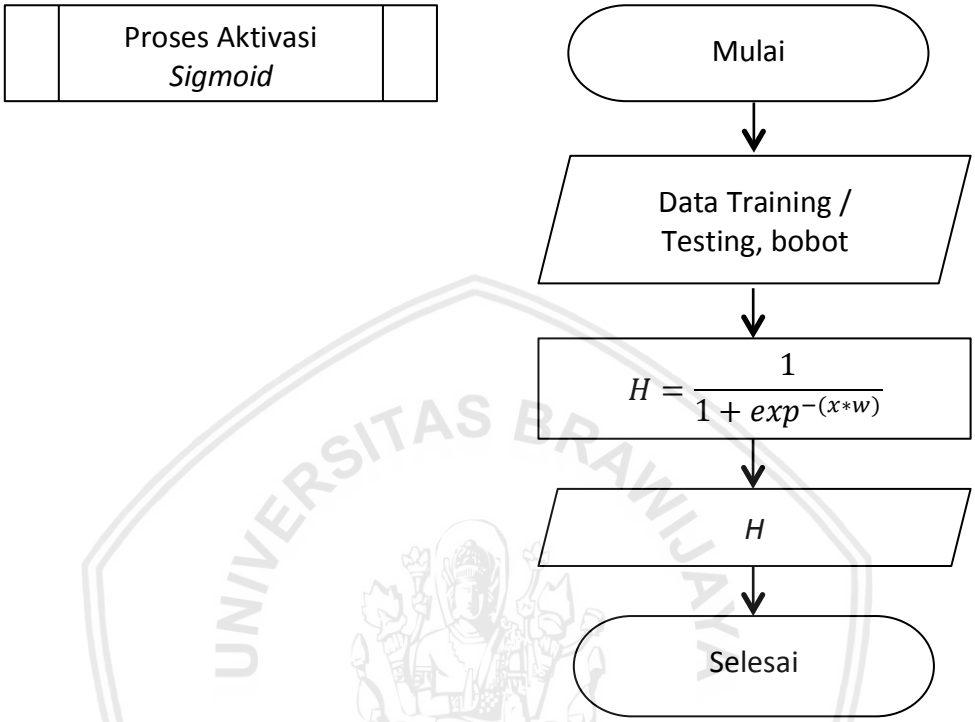
Gambar 4.11 *Flowchart* Proses *Training* ELM

Langkah-langkah proses *training* dapat dijelaskan sebagai berikut:

1. Masukkan data status gizi pada balita, target kelas, dan bobot awal.
2. Melakukan perhitungan aktivasi *output hidden neuron* dengan fungsi aktivasi *sigmoid biner*, melakukan perhitungan matriks *Moore-Penrose*, dan perhitungan *output weight*.
3. Mendapatkan nilai *output weigh*t yang selanjutnya akan digunakan pada proses *testing*.

4.4.3 Proses Aktivasi *Sigmoid*

Proses aktivasi *sigmoid* adalah proses mengaktivasi hasil perkalian antara data dengan bobot. Proses ini dilakukan pada tahap *training* dan *testing*. Alur proses aktivasi dapat dilihat pada Gambar 4.12.



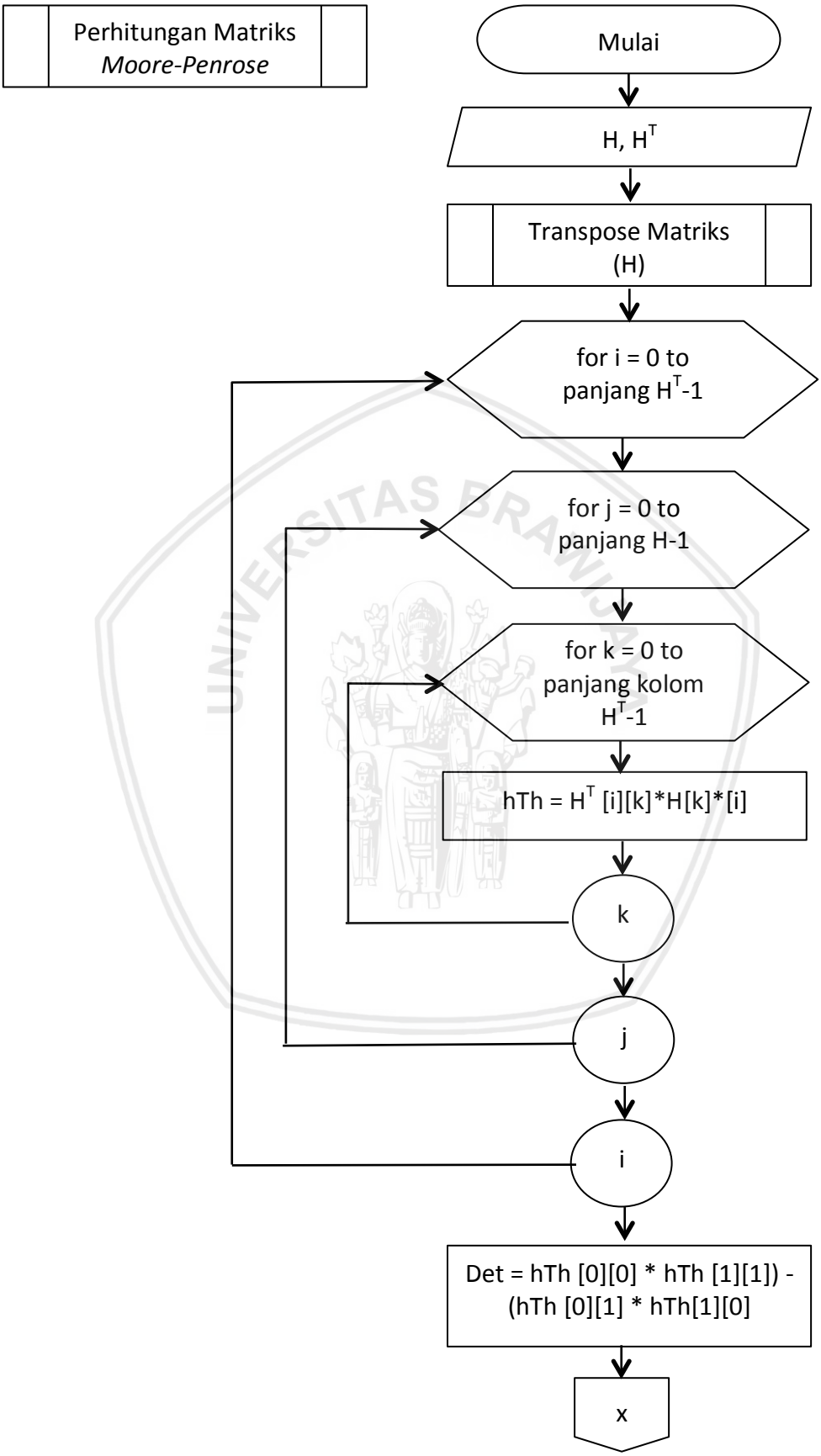
Gambar 4.12 *Flowchart* Proses Aktivasi *Sigmoid*

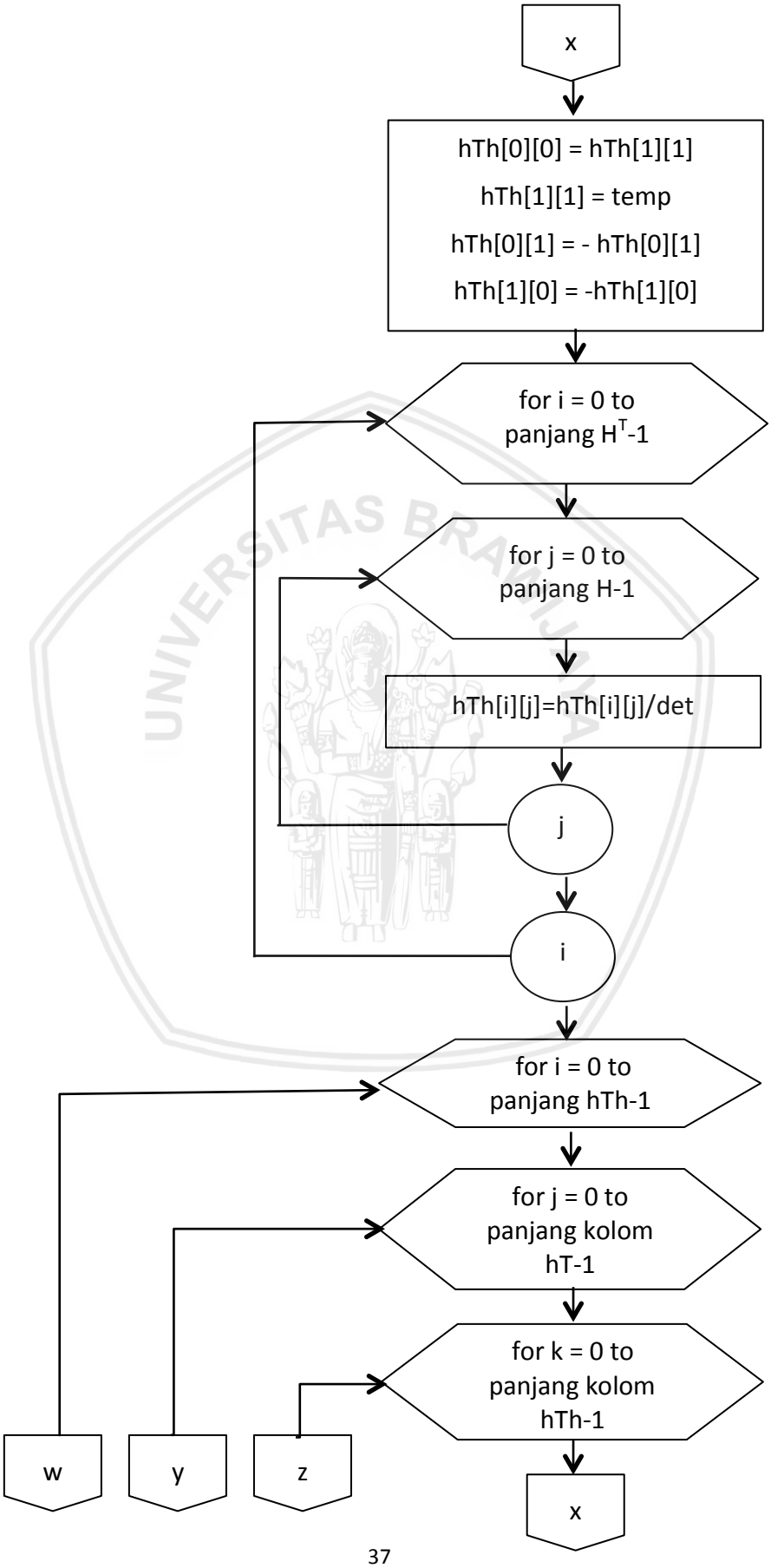
Berdasarkan Gambar 4.12, langkah-langkah perhitungan proses aktivasi *sigmoid* adalah sebagai berikut:

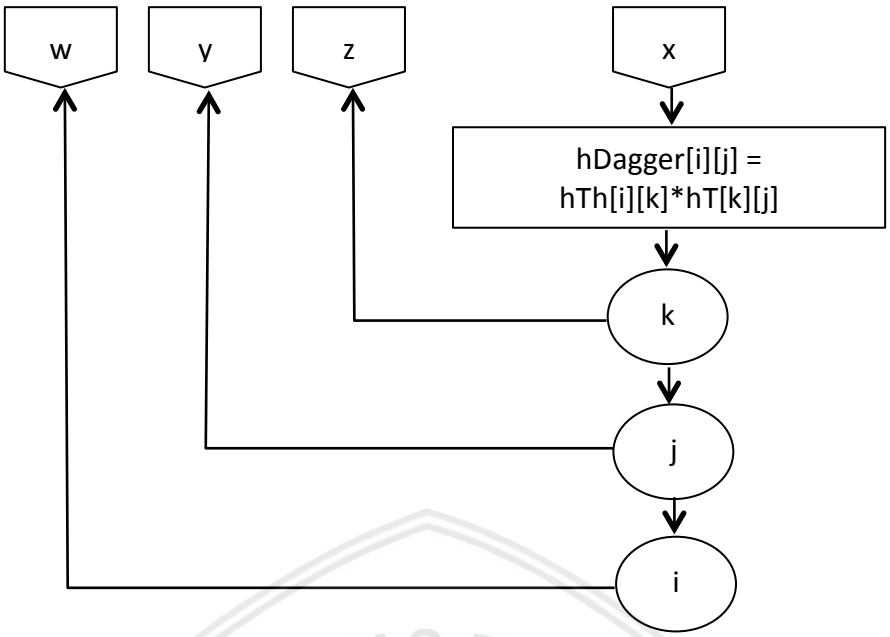
1. Masukkan data *training* atau *testing*, serta bobot.
2. Melakukan perhitungan menggunakan Persamaan (2.3).
3. Mendapatkan nilai matriks *output hidden neuron* teraktivasi.

4.4.4 Proses Perhitungan Matriks *Moore-Penrose*

Berbeda dengan proses perhitungan aktivasi sigmoid, perhitungan matriks *Moore-Penrose* hanya dilakukan pada proses *training*. Alur proses perhitungan matriks *Moore-Penrose* dapat dilihat pada Gambar 4.13.


Perhitungan Matriks
Moore-Penrose
Mulai
H, H^T
Transpose Matriks
(H)
for i = 0 to
panjang H^T-1
for j = 0 to
panjang H-1
for k = 0 to
panjang kolom
H^T-1
hTh = H^T [i][k]*H[k]*[i]
k
j
i
Det = hTh [0][0] * hTh [1][1]) -
(hTh [0][1] * hTh[1][0]
x


x
hTh[0][0] = hTh[1][1]
hTh[1][1] = temp
hTh[0][1] = - hTh[0][1]
hTh[1][0] = -hTh[1][0]
for i = 0 to
panjang H^T-1
for j = 0 to
panjang H-1
hTh[i][j]=hTh[i][j]/det
j
i
for i = 0 to
panjang hTh-1
for j = 0 to
panjang kolom
hT-1
for k = 0 to
panjang kolom
hTh-1
w
y
z
x



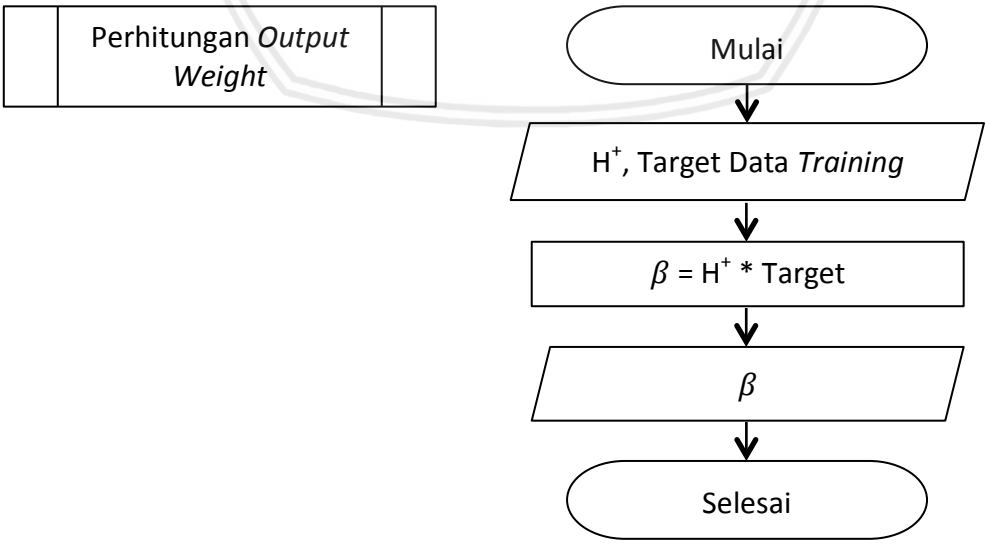
Gambar 4.13 *Flowchart* Perhitungan Matriks *Moore-Penrose*

Penjelasan proses perhitungan matriks Moore-Penrose pada Gambar 4.13 dapar dijabarkan sebagai berikut:

1. Masukkan nilai *output hidden neuron* teraktivasi.
2. Melakukan perhitungan menggunakan Persamaan (2.3).
3. Mendapatkan nilai matriks *Moore-Penrose*.

4.4.5 Proses Perhitungan *Output Weight*

Proses ini hanya dilakukan pada saat *training*, namun nilai yang dihasilkan digunakan untuk proses *testing*. Gambar 4.14 merupakan alur proses perhitungan *output weight*.



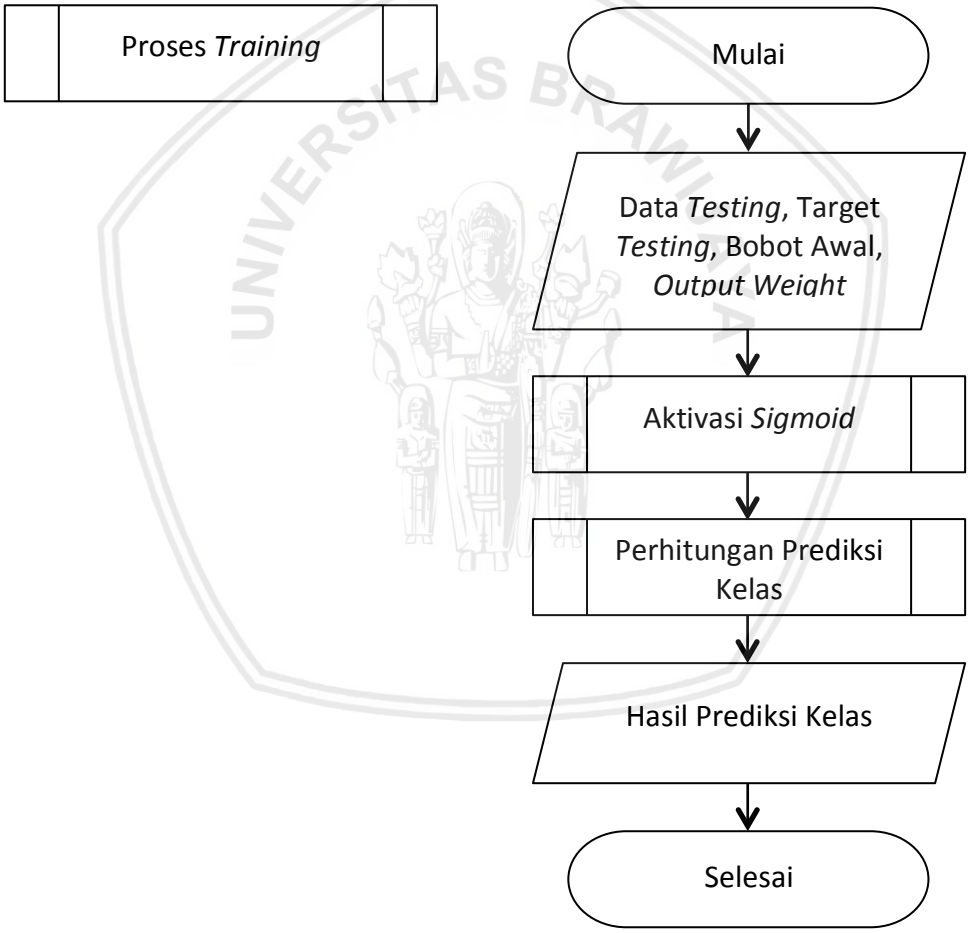
Gambar 4.14 *Flowchart* Perhitungan *Output Weight*

Berdasarkan Gambar 4.14, langkah-langkah perhitungan *output weight* adalah sebagai berikut:

1. Masukkan nilai matriks *Moore-Penrose* dan target data *training*.
2. Melakukan perhitungan menggunakan *output weight* menggunakan Persamaan (2.4).
3. Mendapatkan nilai *output weight*.

4.4.6 Proses *Testing*

Setelah melakukan proses *training*, langkah selanjutnya yaitu proses *testing*. Proses *testing* terdiri dari perhitungan aktivasi *sigmoid* dan perhitungan prediksi kelas. Alur tentang proses *testing* dapat dilihat pada Gambar 4.15.



Gambar 4.15 *Flowchart* Proses *Testing*

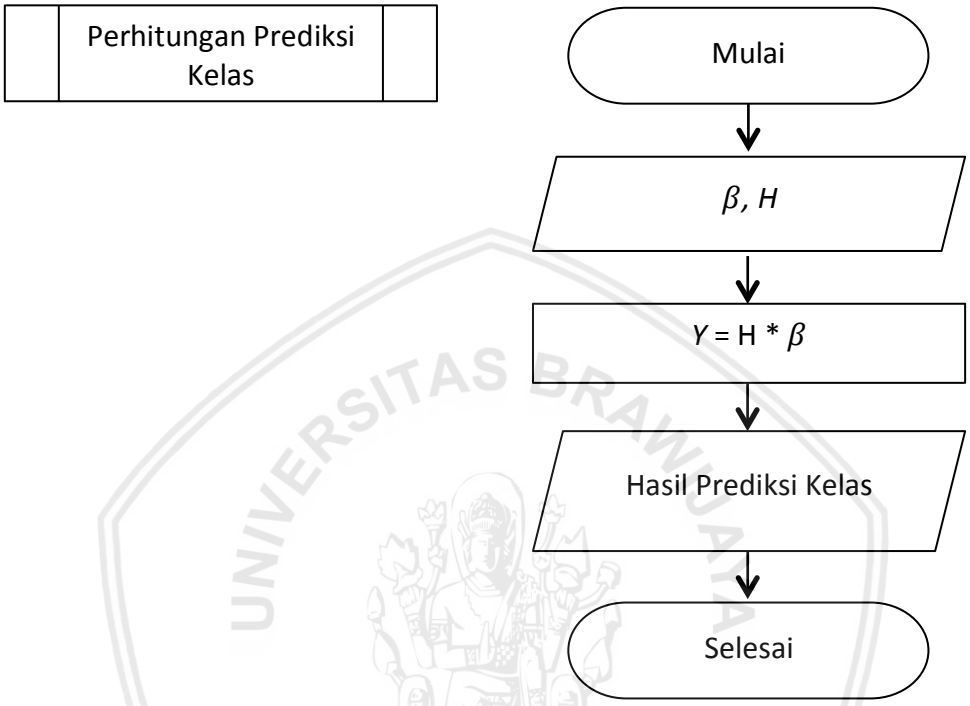
Berdasarkan Gambar 4.15, langkah-langkah proses *testing* adalah sebagai berikut:

1. Masukkan data *testing*, target *testing*, bobot awal, *output weight*.
2. Melakukan perhitungan aktivasi sigmoid menggunakan Persamaan (2.3).
3. Melakukan perhitungan prediksi kelas menggunakan Persamaan (2.5).

4. Mendapatkan hasil prediksi kelas.

4.4.7 Proses Perhitungan Prediksi Kelas

Tahap terakhir yaitu melakukan perhitungan prediksi kelas. Alur tentang proses perhitungan prediksi kelas ditunjukkan pada Gambar 4.16.



Gambar 4.16 *Flowchart* Perhitungan Prediksi Kelas

Penjelasan perhitungan prediksi kelas pada Gambar 4.16 adalah sebagai berikut:

1. Masukkan nilai *output weight* yang didapat dari proses *training* dan nilai *H* dari proses aktivasi *sigmoid* pada proses *testing*.
2. Melakukan perhitungan prediksi kelas menggunakan Persamaan (2.5).
3. Mendapatkan hasil prediksi kelas.

4.5 Perhitungan Manual

Perhitungan manual digunakan untuk memberikan gambaran terkait tahapan algoritme yang digunakan sebelum diimplementasikan ke dalam program. Pada manualisasi ini digunakan 2 *hidden neuron*. Dataset yang digunakan pada proses perhitungan manualisasi ini terdiri dari:

1. 9 data yang digunakan sebagai data *training.*
2. 3 data yang digunakan sebagai data *testing*.
3. Terdapat 4 *input* yang terdiri dari jenis kelamin, umur (bulan), berat badan (kg), dan tinggi badan (cm).

4. Terdapat 3 *output*, yaitu kelas 1, kelas 2, dan kelas 3.

4.5.1 Inisialisasi Kromosom Awal

Nilai kromosom awal pada Algoritme Genetika dibangkitkan secara *random*. Pada penelitian ini, kromosom yang dibangkitkan merupakan bobot yang akan digunakan pada perhitungan untuk klasifikasi menggunakan metode ELM. Oleh karena itu kromosom yang dibangkitkan yaitu memiliki *range* [-1, 1]. Adapun parameter yang digunakan pada Algoritme Genetika adalah sebagai berikut:

- *popSize* = 4
- *cr* = 0.4
- *mr* = 0.2

Karena *popSize* bernilai 4, maka didapatkan 4 individu yang telah dibangkitkan secara *random*. Setiap individu terdiri dari 8 gen yang terdiri dari 8 bobot yang akan digunakan sebagai bobot awal pada metode ELM.

Tabel 4.2 Inisialisasi Kromosom Awal

P_1	0.557	0.834	0.234	0.923	0.419	0.734	0.962	0.653
P_2	0.982	0.233	0.821	0.231	0.350	0.853	0.478	0.553
P_3	-0.332	-0.511	0.756	0.962	0.458	0.287	0.971	0.923
P_4	0.238	0.794	0.321	0.827	0.723	0.856	0.172	0.952

4.5.2 Proses Reproduksi

Setelah kromosom awal telah diinisialisasi, maka dilakukan proses reproduksi untuk mendapatkan *offspring*. Seperti yang telah disebutkan sebelumnya bahwa proses reproduksi terdiri atas dua bagian, yaitu *crossover* dan mutasi.

4.5.2.1 Crossover

Pada *crossover*, *offspring* yang dihasilkan sebanyak *cr* x *popSize*, yaitu 0.4 x 4 = 1.6 $\approx$ 2 sehingga proses *crossover* hanya dilakukan sekali. Kemudian karena metode *crossover* yang digunakan yaitu *Extended Intermediate Crossover,* maka diperlukan variabel tambahan untuk membantu perhitungan yaitu variabel *alpha* (α). Variabel α dibangkitkan secara *random* dengan *range* [-0.25, 1.25]. Berikut merupakan contoh hasil pembangkitan variabel α.

Tabel 4.3 Pembangkitan Nilai *Alpha*

α	0.726	0.423	0.239	-0.192	1.182	0.287	0.312	0.123

Setelah didapatkan nilai α, kemudian dilakukan pemilihan 2 *parent* yang akan digunakan untuk melakukan proses *crossover*. Berikut contoh perhitungan *crossover* menggunakan *Extended Intermediate Crossover* dengan melibatkan P_1 dan P_2 sebagai *parent* menggunakan Persamaan (2.6).

C_1 = 0.557 + 0.726(0.982 – 0.557)

= 0.866

C_2 = 0.982 + 0.726(0.557 – 0.982)

= 0.673

Hasil *crossover* dapat dilihat pada Tabel 4.4.

Tabel 4.4 Hasil Proses *Crossover* P_1 dan P_2

C_1	0.866	0.580	0.374	1.056	0.337	0.768	0.811	0.641
C_2	0.673	0.487	0.681	0.098	0.432	0.819	0.629	0.565

4.5.2.2 Mutasi

Offspring yang dihasilkan pada proses mutasi sebanyak *mr* x *popSize*, yaitu 0.2 x 4 = 0.8 ≈ 1. Karena metode mutasi yang digunakan adalah *random mutation*, maka dibutuhkan satu variabel tambahan untuk membantu proses perhitungan, yaitu variabel *r* yang dibangkitkan secara random pada *range* [-0.1, 0.1]. Dimisalkan nilai *r* adalah 0.067 dan individu yang digunakan adalah P4 indeks 4. Maka contoh perhitungan mutasi menggunakan Persamaan (2.7) adalah sebagai berikut:

C_3 = 0.827 + 0.067(1 - (-1))

= 0.961

Hasil proses mutasi dapat dilihat pada Tabel 4.5.

Tabel 4.5 Hasil Proses Mutasi P_4

C_3	0.238	0.794	0.321	0.961	0.723	0.856	0.172	0.952

Setelah dilakukan proses *crossover* dan mutasi, adapun individu gabungan yang dihasilkan oleh proses reproduksi dapat dilihat pada Tabel 4.6.

Tabel 4.6 Individu Gabungan Hasil Reproduksi

P_1	0.557	0.834	0.234	0.923	0.419	0.734	0.962	0.653
P_2	0.982	0.233	0.821	0.231	0.350	0.853	0.478	0.553
P_3	-0.332	-0.511	0.756	0.962	0.458	0.287	0.971	0.923
P_4	0.238	0.794	0.321	0.827	0.723	0.856	0.172	0.952
C_1	0.866	0.580	0.374	1.056	0.337	0.768	0.811	0.641
C_2	0.673	0.487	0.681	0.098	0.432	0.819	0.629	0.565
C_3	0.238	0.794	0.321	0.961	0.723	0.856	0.172	0.952

4.5.3 Perhitungan *Extreme Learning Machine*

Setelah mendapatkan *offspring*, langkah selanjutnya yaitu mencari nilai *fitness* masing-masing individu dengan menggunakan metode ELM.

4.5.3.1 Normalisasi Data

Pada ELM, data dibagi menjadi dua macam, yaitu data *training* dan *testing*. Terdapat 4 parameter *input*, yaitu jenis kelamin yang dilambangkan dengan X_1, berisi bilangan 1 dan 0 dimana 1 untuk laki-laki dan 0 untuk perempuan. Parameter selanjutnya yaitu umur yang dilambangkan dengan X_2, berat badan yang dilambangkan dengan X_3, dan tinggi badan yang dilambangkan dengan X_4. Data yang digunakan adalah data sampel yang terdapat pada Tabel 4.1. Pembagian data tersebut dapat dilihat pada Tabel 4.7 untuk data *training* dan Tabel 4.8 untuk data *testing*.

Tabel 4.7 Data *Training*

No.	X_1	X_2	X_3	X_4	Kelas
1.	1	34	20.5	96	Kelas 1
2.	1	55	29	110	Kelas 1
3.	0	36	22	97	Kelas 1
4.	1	41	17	95	Kelas 2
5	1	56	2.2	107	Kelas 2
6.	1	20	14	84	Kelas 2
7.	1	23	10.2	79	Kelas 3
8.	0	23	9	78	Kelas 3
9.	0	17	10	73	Kelas 3

Tabel 4.8 Data *Testing*

No.	X_1	X_2	X_3	X_4	Kelas
1.	1	59	31.5	112	Kelas 1
2.	0	56	19	102	Kelas 2
3.	0	33	12	82	Kelas 3

Sebelum masuk ke proses *training* dan *testing*, dilakukan normalisasi data terlebih dahulu untuk mengurangi perbedaan rentang data. Berikut contoh perhitungan normalisasi data.

$$X_1 = \frac{1-0}{1-0} = 1$$

$$X_2 = \frac{34-17}{59-17} = 0.405$$

$$X_3 = \frac{20.5-2.2}{31.5-2.2} = 0.625$$

$$X_4 = \frac{96 - 73}{113 - 73} = 0.590$$

Keseluruhan hasil normalisasi data dapat dilihat pada Tabel 4.9 dan Tabel 4.10.

Tabel 4.9 Normalisasi Data *Training*

No.	X_1	X_2	X_3	X_4	Kelas
1.	1.000	0.405	0.625	0.590	Kelas 1
2.	1.000	0.905	0.915	0.949	Kelas 1
3.	0.000	0.452	0.676	0.615	Kelas 1
4.	1.000	0.571	0.505	0.564	Kelas 2
5	1.000	0.929	0.000	0.872	Kelas 2
6.	1.000	0.071	0.403	0.282	Kelas 2
7.	1.000	0.143	0.273	0.154	Kelas 3
8.	0.000	0.143	0.232	0.128	Kelas 3
9.	0.000	0.000	0.266	0.000	Kelas 3

Tabel 4.10 Normalisasi Data *Testing*

No.	X_1	X_2	X_3	X_4	Kelas
1.	1.000	1.000	1.000	1.000	Kelas 1
2.	0	0.928571429	0.573	0.744	Kelas 2
3.	0	0.380952381	0.334	0.231	Kelas 3

4.5.3.2 Proses Training

1. Inisialisasi Bobot Awal

   Langkah pertama yang dilakukan adalah melakukan inisialisasi parameter yang terdiri dari bobot awal yang dioptimasi menggunakan Algoritme Genetika serta data status gizi yang telah dinormalisasi. Bobot awal yang digunakan diambil dari salah satu individu gabungan (lihat Tabel 4.6). Ukuran bobot yang dibutuhkan adalah 2x4. Oleh karena itu, individu gabungan perlu diubah dari matriks 1x8 menjadi 2x4. Sebagai contoh, individu yang dijadikan bobot awal adalah P_1 yang dapat dilihat pada Tabel 4.11 dan hasil transformasinya ditunjukkan pada Tabel 4.12.

Tabel 4.11 Individu P_1 Sebagai Bobot Awal

P_1	0.557	0.834	0.234	0.923	0.419	0.734	0.962	0.653

Tabel 4.12 Transformasi Individu P_1

w	1	2	3	4
1	0.557	0.834	0.234	0.923
2	0.419	0.734	0.962	0.653

2. Menghitung Matriks Aktivasi *Output Hidden Neuron*

Untuk menghitung matriks aktivasi *output hidden neuron* menggunakan fungsi *sigmoid* biner, digunakan Persamaan (2.2). Data *input* yang digunakan merupakan data *training* dengan ordo 9 x 4 dikalikan dengan bobot awal yang di-*transpose* yang dapat dilihat pada Tabel 4.13.

Tabel 4.13 Bobot *Transpose*

w^T	1	2
1	0.557	0.419
2	0.834	0.734
3	0.234	0.962
4	0.923	0.653

Langkah selanjutnya yaitu melakukan aktivasi menggunakan aktivasi *sigmoid biner*. Dengan menggunakan Persamaan (2.2), contoh perhitungannya adalah sebagai berikut:

$$H_{1,1} = \frac{1}{1 + exp^{-(1*0.557+0.405*0.834+0.625*0.234+0.590*0.923)}}$$
$$= 0.830$$

Hasil aktivasi *output hidden neuron* ditunjukkan pada Tabel 4.14.

Tabel 4.14 Hasil Aktivasi *Sigmoid*

H	1	2
1.	0.830	0.846
2.	0.917	0.930
3.	0.751	0.800
4.	0.842	0.845
5.	0.894	0.842
6.	0.725	0.739
7.	0.707	0.708
8.	0.572	0.602
9.	0.516	0.564

3. Mencari matriks *Moore-Penrose Pseudo Inverse*

Langkah selanjutnya yaitu menghitung matriks *Moore-Penrose Pseudo Inverse* menggunakan Persamaan (2.3). Pertama lakukan *transpose* pada matriks yang telah teraktivasi, yang ditunjukkan pada Tabel 4.15.

Tabel 4.15 Matriks *H* Transpose (H^T)

H^T	1	2	3	4	5	6	7	8	9
1	0.830	0.917	0.751	0.842	0.894	0.725	0.707	0.572	0.516
2	0.846	0.930	0.800	0.845	0.842	0.739	0.708	0.602	0.564

Setelah itu, kalikan H^T dengan matriks *H* sehingga akan menghasilkan matriks dengan ordo 2x2. Contoh perhitungannya adalah sebagai berikut:

$(H^TH)_{1,1}$ = 0.830*0.830 + 0.917*0.917 + 0.751*0.751 + 0.842*0.842 + 0.894*0.894 + 0.725*0.725 + 0.707*0.707 + 0.572*0.572 + 0.516*0.516

= 5.222

Keseluruhan hasil perkalian H^T dan *H* dapat dilihat pada Tabel 4.16.

Tabel 4.16 Hasil Perkalian Matriks H^T dan *H*

H^TH	1	2
1	5.222	5.291
2	5.291	5.369

Kemudian cari *invers* dari hasil perkalian tersebut sehingga didapatkan matriks berikut:

$(H^TH)^{-1}$	1	2
1	127.928	-126.072
2	-126.072	124.429

Setelah didapatkan nilai *invers* dari matriks, kalikan hasil invers tersebut *($(H^TH)^{-1}$)* dengan nilai *H* yang telah di-*transpose* *(H^T)* maka akan didapatkan *Matriks Moore-Pseudo Inverse* seperti yang ditunjukkan pada Tabel 4.17.

Tabel 4.17 Matriks *Moore-Penrose Pseudo Inverse*

H^+	1	2	3	4	5	6	7	8	9
1	-0.462	0.088	-4.750	1.225	8.317	-0.430	1.184	-2.606	-5.108
2	0.613	0.087	4.830	-1.050	-8.040	0.561	-1.035	2.680	5139

Sebelum melakukan perhitungan *output weight*, matriks target perlu diubah seperti contoh pada Tabel Tabel 4.18. Target awal ke-*i* akan bernilai 1 pada kolom *t(i)*, sedangkan lainnya bernilai -1. Kemudian dilakukan perhitungan *output weight*.

Tabel 4.18 Target *Training*

Data	Target		
	t1	t2	t3
1.	1	-1	-1
2.	1	-1	-1
3.	1	-1	-1
4.	-1	1	-1
5.	-1	1	-1
6.	-1	1	-1
7.	-1	-1	1
8.	-1	-1	1
9.	-1	-1	1

4. Hitung *Output Weight*

Langkah selanjutnya yaitu menghitung *output weight* menggunakan Persamaan (2.4). Sehingga akan didapatkan matriks seperti pada Tabel 4.19.

Tabel 4.19 Hasil *Output Weight*

β	1	2	3
1	-7.706	20.767	-10.518
2	7.273	-20.842	9.783

4.5.3.3 Proses Testing

1. Melakukan aktivasi matriks *(H)*

Mencari nilai *H* yaitu dengan menggunakan fungsi aktivasi yang terdapat pada Persamaan (2.3). Contoh perhitungan aktivasi matriks menggunakan fungsi *sigmoid* biner adalah sebagai berikut:

$$H_{1,1} = \frac{1}{1 + exp^{-(1*0.557+1*0.834+1*0.234+1*0.923)}}$$
$$= 0.927$$

Hasil aktivasi *output hidden neuron* ditunjukkan pada Tabel 4.20.

Tabel 4.20 Hasil Aktivasi *Sigmoid*

H	1	2
1	0.927	0.941
2	0.831	0.848
3	0.648	0.680

2. Menghitung Hasil *Output*

Hasil *output* didapatkan dengan melakukan perkalian matriks hasil aktivasi *sigmoid* data *testing* dengan matriks *output weight* yang didapatkan pada proses *training* (lihat Tabel 4.19) menggunakan Persamaan (2.5). Berikut merupakan contoh perhitungannya.

$Y_{1,1}$ = (0.927*(-7.706)) + (0.941*(7.273))
= -0.3034

Keseluruhan perhitungan hasil *output* ditunjukkan pada Tabel 4.21.

Tabel 4.21 Hasil *Output*

Y	1	2	3
1	-0.303	-0.351	-0.550
2	-0.239	-0.410	-0.448
3	-0.048	-0.714	-0.164

Setelah *output* telah didapatkan, hasil prediksi dapat diketahui dengan menghitung nilai maksimal dari masing-masing *output*. Hasil prediksi kelas ditunjukkan pada Tabel 4.22.

Tabel 4.22 Hasil Prediksi Kelas

Data *Testing*	*Output* 1	*Output* 2	*Output* 3	Nilai Maksimal	Prediksi Kelas	Kelas Aktual
1	-0.303	-0.351	-0.550	-0.303	1	1
2	-0.239	-0.410	-0.448	-0.239	1	2
3	-0.048	-0.714	-0.164	-0.048	1	3

4.5.4 Evaluasi

Pada proses evaluasi dilakukan perhitungan nilai *fitness* dengan cara menghitung akurasi dari hasil klasifikasi menggunakan Persamaan (2.8). Berikut merupakan contoh dari perhitungan nilai *fitness* dari data pada Tabel 4.22.

$$Akurasi\ Prediksi\ ELM = \frac{1}{3} \times 100\% = 33.33\%$$

$$f(x) = Akurasi\ PrediksiELM(x)$$

$$f(P_1) = 33.33$$

Setelah dilakukan perhitungan prediksi kelas menggunakan nilai dari individu gabungan secara bergantian, maka didapatkan nilai *fitness* seluruh individu sebagai berikut:

Tabel 4.23 Nilai *Fitness* Individu Gabungan

Individu	*Fitness*
P_1	33.33
P_2	33.33
P_3	33.33
P_4	33.33
C_1	33.33
C_2	33.33
C_3	33.33

4.5.5 Seleksi

Setelah dilakukan evaluasi, dilakukan proses seleksi dengan cara mengurutkan nilai *fitness* secara *descending* agar memudahkan dalam memilih individu terbaik yang lolos dan dihitung kembali pada iterasi berikutnya. Berikut contoh hasil pengurutan individu.

Tabel 4.24 Pengurutan Nilai *Fitness*

Individu	*Fitness*
P_1	33.33
P_2	33.33
P_3	33.33
P_4	33.33
C_1	33.33
C_2	33.33
C_3	33.33

Setelah diurutkan (pada contoh, nilai *fitness* sama sehingga langsung terurut), individu tersebut kemudian diambil sejumlah nilai *popSize* teratas untuk diproses kembali pada iterasi berikutnya. Individu yang memiliki nilai *fitness* tertinggi akan menjadi bobot awal yang terbaik. Karena *popSize* bernilai 4, maka individu yang lolos ke generasi selanjutnya adalah sebagai berikut:

Tabel 4.25 Hasil Seleksi Individu

Individu	*Fitness*
P_1	33.33

P_2	33.33
P_3	33.33
P_4	33.33

4.6 Perancangan Pengujian

Pada penelitian ini dilakukan pengujian-pengujian yang berkaitan dengan metode *Extreme Learning Machine* dan Algoritme Genetika yang digunakan untuk mengklasifikasikan status gizi pada balita. Pengujian dilakukan menggunakan 6 skenario pengujian, dimana setiap skenario memiliki tujuan untuk mengukur kinerja algoritme berdasarkan parameter yang digunakan. 4 skenario pengujian yang digunakan adalah pengujian konvergensi, pengujian ukuran populasi, pengujian pengaruh kombinasi nilai *cr* dan *mr*, dan pengujian jumlah *hidden neuron*.

4.6.1 Perancangan Pengujian Konvergensi

Pengujian konvergensi digunakan untuk mengetahui pada iterasi berapa nilai *fitness* konvergen dan stagnan. Perancangan pengujian jumlah generasi dapat dilihat pada Tabel 4.26.

Tabel 4.26 Skenario Pengujian Konvergensi

Jumlah Iterasi	Akurasi				
	Percobaan ke-				
	1	2	3	4	5
1					
2					
3					
4					
5					
6					
...					
200					

4.6.2 Perancangan Pengujian Ukuran Populasi

Pengujian ukuran populasi digunakan untuk menentukan *popsize* terbaik untuk menghasilkan solusi terbaik. Pengujian ukuran populasi dilakukan dengan melakukan pengubahan nilai ukuran populasi sebanyak kelipatan 5 yang dimulai dari 5 hingga 50. Perancangan pengujian ukuran populasi dapat dilihat pada Tabel Tabel 4.27.

Tabel 4.27 Skenario Pengujian Ukuran Populasi

Ukuran Populasi	Akurasi					Rata-Rata Akurasi (%)
	Percobaan ke-					
	1	2	3	4	5	
10						
20						
30						
40						
50						
60						
70						
80						
90						
100						

4.6.3 Perancangan Pengujian Pengaruh Kombinasi Nilai *Crossover Rate* dan *Mutation Rate*

Pada penelitian ini akan dilakukan pengujian nilai *crossover rate (cr)* dan *mutation rate (mr)* untuk mengetahui apakah kombinasi nilai *cr* dan *mr* memengaruhi akurasi sistem, serta mengetahui nilai kombinasi yang dapat menghasilkan solusi terbaik. Skenario pengujian pengaruh kombinasi nilai *cr* dan *mr* ditunjukkan pada Tabel 4.28.

Tabel 4.28 Skenario Pengujian Nilai *Cr* dan *Mr*

Nilai Kombinasi		Akurasi (%)					Rata-Rata Akurasi (%)
		Percobaan ke-					
cr	*mr*	1	2	3	4	5	
1	0						
0.9	0.1						
0.8	0.2						
0.7	0.3						
0.6	0.4						
0.5	0.5						
0.4	0.6						

0.3	0.7						
0.2	0.8						
0.1	0.9						
0	1						

4.6.4 Perancangan Pengujian Jumlah *Hidden Neuron*

Pengujian jumlah *hidden neuron* dilakukan dengan melakukan percobaan jumlah *hidden neuron* yang diharapkan mampu menjadi parameter yang optimal pada penelitian ini. Pengujian jumlah *hidden neuron* pada penelitian ini terbagi menjadi 2, yakni terhadap akurasi dan waktu. Pengujian jumlah *hidden neuron* terhadap akurasi dilakukan untuk mengetahui nilai akurasi dari sistem. Pengujian jumlah *hidden neuron* terhadap waktu dilakukan untuk mengetahui lama sistem melakukan proses perhitungan. Perancangan pengujian jumlah *hidden neuron* dapat dilihat pada Tabel 4.29.

Tabel 4.29 Skenario Pengujian Jumlah *Hidden Neuron*

Percobaan ke-	Jumlah *Hidden Neuron*									
	1		2		3		4		5	
	A	W	A	W	A	W	A	W	A	W
1										
2										
3										
4										
5										
Rata-Rata										

Keterangan: A = akurasi, W = waktu

BAB 5 IMPLEMENTASI

Bab implementasi membahas tentang spesifikasi perangkat yang digunakan dalam penelitian serta penjelasan mengenai implementasi sistem yang dibangun.

5.1 Spesifikasi Sistem

Spesifikasi sistem digunakan untuk mengetahui spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam melakukan implementasi sistem pada penelitian ini.

5.1.1 Spesifikasi Perangkat Keras

Perangkat keras yang digunakan pada penelitian ini adalah sebagai berikut:

- Processor Intel Core i5 2.7 GHz
- RAM 8 GB DDR3
- Monitor 13-inch

5.1.2 Spesifikasi Perangkat Lunak

Perangkat lunak yang digunakan pada penelitian ini adalah sistem operasi MacOS Sierra 10.12.6.

5.2 Implementasi Program

Pada implementasi program terdapat pembahasan implementasi mengenai dua metode yang digunakan, yaitu Algoritme Genetika dan ELM. Pada Algoritme Genetika terdapat 4 tahapan proses perhitungan, yaitu proses inisialisasi populasi awal, reproduksi (*crossover* dan mutasi), serta evaluasi dan seleksi. Sedangkan pada ELM terdapat 8 tahapan, meliputi normalisasi data, inisialisasi bobot awal, perhitungan fungsi aktivasi, *output weight*, *output layer,* dan hasil prediksi. Implementasi ini menggunakan bahasa pemrograman Java dengan editor Neatbeans IDE 8.0.2, serta *library* JXL untuk mengakses *file excel*.

5.2.1 Implementasi Inisialisasi Populasi Awal

Inisialisasi populasi awal dilakukan secara *random* pada *range* [-1, 1] dengan jumlah populasi sebanyak nilai *popsize*. Implementasi inisialisasi populasi awal dapat dilihat pada Kode Sumber 5.1.

Kode Sumber 5.1 Implementasi Inisialisasi Populasi Awal

```
Algoritme 1: Fungsi Inisialisasi Populasi Awal
1   double[][] inisialisasi() {
2       Random rand = new Random();
3       double max = 1, min = -1;
4       for (int i = 0; i < popsize; i++) {
5           for (int j = 0; j < gen; j++) {
6               bil[i][j] = (rand.nextDouble() * ((max - min))) + min;
7           }
8       }
9       return bil;
10  }
```

Penjelasan Kode Sumber 5.1 tentang implementasi inisialisasi populasi awal adalah sebagai berikut:

Baris 2 : Membuat objek bernama *rand* dari kelas *Random*.

Baris 3 : Inisialisasi variabel *max* dan *min.*

Baris 7 : Proses membangkitkan nilai populasi awal secara *random* pada rentang nilai variabel *min* dan *max*, yakni -1 dan 1.

Baris 9 Mengembalikan nilai dari variabel *bil.*

5.2.2 Implementasi Proses Reproduksi

Terdapat dua proses dalam reproduksi, yaitu *crossover* dan mutasi. Implementasi kedua proses tersebut dapat dilihat pada sub bab berikut:

5.2.2.1 Implementasi Proses Crossover

Pada penelitian ini digunakan *Extended Intermediate Crossover* sebagai metode *crossover*-nya. Dalam prosesnya dibutuhkan dua *parent* dan variabel *alpha* untuk menghasilkan dua *offspring*. Nilai *alpha* dibangkitkan secara acak pada *range* [-0.25, 1.25]. Implementasi proses *crossover* dapat dilihat pada Kode Sumber 5.2.

Kode Sumber 5.2 Implementasi Proses *Crossover*

Algoritme 2: Fungsi Proses *Crossover (Extended Intermediate Crossover)*

```
1  double[][] crossover(double[][] bil) {
2      double[] alpha = new double[gen];
3      int jumlahParent;
4      int jumlahChild = (int) Math.ceil(cr * popsize);
5      double child[][] = new double[jumlahChild][gen];
6      double tempIndividu[][] = new double[bil.length +
                                jumlahChild][gen];
7      int counterCrossover = bil.length;
8
9      if (jumlahChild % 2 == 0) {
10         jumlahParent = jumlahChild;
11     } else {
12         jumlahParent = jumlahChild + 1;
13     }
14
15     double parentTerpilih[][] = new double[jumlahParent][gen];
16     int min = 0, max = popsize;
17
18     ArrayList<Integer> list = new ArrayList<Integer>();
19     for (int i = min; i < max; i++) {
20         list.add(new Integer(i));
21     }
22
23     Collections.shuffle(list);
24
25     for (int i = 0; i < jumlahParent; i++) {
26         for (int j = 0; j < gen; j++) {
27             if (j == (gen - 1)) {
28                 parentTerpilih[i][j] = bil[list.get(i)][j];
29             } else {
30                 parentTerpilih[i][j] = bil[list.get(i)][j];}
31             }
32         }
```

```
33      }
34
35      for (int i = 0; i < gen; i++) {
36          Random rand = new Random();
37          double minAlpha = -0.25, maxAlpha = 1.25;
38          alpha[i] = (rand.nextDouble() * (maxAlpha - minAlpha)) +
                    minAlpha;
39      }
40
41      for (int i = 0; i < jumlahChild; i++) {
42          for (int j = 0; j < gen; j++) {
43              if (i == 0 || i % 2 == 0) {
44                  child[i][j] = parentTerpilih[i][j] + (alpha[j] *
                                (parentTerpilih[i + 1][j] -
                                 parentTerpilih[i][j]));
45                  tempIndividu[counterCrossover][j] = child[i][j];
46              } else {
47                  child[i][j] = parentTerpilih[i][j] + (alpha[j] *
                                (parentTerpilih[i - 1][j] -
                                 parentTerpilih[i][j]));
48                  tempIndividu[counterCrossover][j] = child[i][j];
49              }
50          }
51          counterCrossover++;
52      }
53      return tempIndividu;
54  }
```

Penjelasan Kode Sumber 5.2 tentang implementasi proses *crossover* adalah sebagai berikut:

Baris 2 s.d. 7 : Deklarasi variabel-variabel yang digunakan dalam proses perhitungan *crossover*.

Baris 9 s.d. 13 : Proses perhitungan jumlah *parent* yang dibutuhkan.

Baris 15 s.d. 16 : Deklarasi dan inisialisasi variabel yang digunakan dalam proses perhitungan *crossover*.

Baris 18 s.d. 33 : Proses pemilihan parent yang akan digunakan serta menyimpan nilai parent tersebut ke dalam variabel.

Baris 35 s.d. 39 : Proses pembangkitan nilai *alpha* dengan *range* [-0.25, 1.25]

Baris 41 s.d. 52 : Proses perhitungan *Extended Intermediate Crossover*, kemudian hasilnya disimpan dalam variabel.

Baris 53 : Mengembalikan nilai dari variabel *tempIndividu*.

5.2.2.2 Implementasi Proses Mutasi

Setelah *crossover*, proses selanjutnya yaitu mutasi. Metode mutasi yang digunakan adalah *Random Mutation*. Metode ini membutuhkan variabel tambahan, yaitu *r* yang dibangkitkan secara *random* pada *range* [-0.1, 0.1]. Implementasi proses mutasi dapat dilihat pada Kode Sumber 5.3.

Kode Sumber 5.3 Implementasi Proses Mutasi

```
Algoritme 3: Fungsi Proses Mutasi (Random Mutation)
1   double[][] mutasi(double[][] x) {
2       int index = (int) (Math.random() * gen);
```

```
3       int jumlahMutasi = (int) (Math.ceil(mr * popsize));
4       double childMutasi[][] = new double[jumlahMutasi][gen];
5       double tempIndividu[][] = new double[x.length +
                                    jumlahMutasi][gen];
6       int counterMutasi = x.length;
7
8       for (int i = 0; i < x.length; i++) {
9           for (int j = 0; j < gen; j++) {
10              tempIndividu[i][j] = x[i][j];
11          }
12      }
13
14      for (int i = 0; i < jumlahMutasi; i++) {
15          double minR = -0.1, maxR = 0.1;
16          double r = (rand.nextDouble() * (maxR - minR)) + minR;
17          int parent = (int) (0 + Math.random() * popsize);
18          for (int j = 0; j < gen; j++) {
19              childMutasi[i][j] = bil[parent][j];
20              if (j == index) {
21                  bil[parent][index] = bil[parent][index] + (r * (1 -
                                            (-1)));
22                  childMutasi[i][j] = bil[parent][j];
23              }
24              tempIndividu[counterMutasi][j] = childMutasi[i][j];
25          }
26          counterMutasi++;
27      }
28      return tempIndividu;
29  }
```

Penjelasan Kode Sumber 5.3 tentang implementasi proses mutasi adalah sebagai berikut:

Baris 2 s.d. 6 : Deklarasi variabel-variabel yang digunakan dalam proses perhitungan mutasi.

Baris 8 s.d. 12 : Proses untuk menyimpan nilai individu gabungan awal.

Baris 14 s.d. 27 : Proses perhitungan mutasi menggunakan *Random Mutation*, yang kemudian disimpan dalam variabel.

Baris 28 : Mengembalikan nilai dari variabel *tempIndividu*.

5.2.3 Implementasi Normalisasi Data

Sebelum data diproses, maka harus dilakukan normalisasi agar semua data berada pada *range* yang sama. Metode yang digunakan untuk melakukan normalisasi adalah *Min – Max Normalization*. Implementasi normalisasi data ditunjukkan pada Kode Sumber 5.4.

Kode Sumber 5.4 Implementasi Normalisasi Data

```
Algoritme 4: Fungsi Proses Normalisasi Data
1   double[][] normalisasi(double x[][]) {
2       double max[] = maxValue(x);
3       double min[] = minValue(x);
4
5       for (int row = 0; row < x.length; row++) {
6           for (int col = 0; col < x[0].length; col++) {
7               x[row][col] = Math.abs(x[row][col] - min[col]) /
                              Math.abs(max[col] - min[col]);
8               x[row][col] = x[row][col];
```

```
9           }
10      }
11          return x;
12  }
```

Penjelasan Kode Sumber 5.4 tentang perhitungan proses normalisasi adalah sebagai berikut:

Baris 2 : Inisialisasi variabel *max* berisi nilai yang didapatkan dari pemanggilan fungsi *maxValue()*.

Baris 3 : Inisialisasi variabel *min* berisi nilai yang didapatkan dari pemanggilan fungsi *minValue()*.

Baris 5 s.d. 11 : Proses perhitungan normalisasi menggunakan metode *Min – Max Normalization*.

Baris 12 : Mengembalikan nilai dari variabel *x*.

5.2.4 Implementasi Perhitungan Fungsi Aktivasi

Fungsi aktivasi yang digunakan adalah *Sigmoid Biner* dengan implementasi seperti pada Kode Sumber 5.5.

Kode Sumber 5.5 Perhitungan Fungsi Aktivasi

```
Algoritme 5: Fungsi Perhitungan Fungsi Aktivasi
1   double[][] aktivasiSigmoidBiner(double[][] hinit) {
2       double[][] h = new double[hinit.length][hinit[0].length];
3       for (int i = 0; i < hinit.length; i++) {
4           for (int j = 0; j < hinit[0].length; j++) {
5               h[i][j] = (1 / (1 + Math.exp(-hinit[i][j])));
6           }
7       }
8       return h;
9   }
```

Penjelasan Kode Sumber 5.5 tentang perhitungan fungsi aktivasi adalah sebagai berikut:

Baris 2 : Deklarasi variabel *h* yang digunakan untuk menyimpan hasil perhitungan.

Baris 3 s.d. 7 : Proses perhitungan fungsi aktivasi menggunakan *Sigmoid Biner*.

Baris 8 : Mengembalikan nilai dari variabel *h*.

5.2.5 Implementasi Perhitungan Matriks *Moore – Penrose*

Implementasi perhitungan matriks *Moore – Penrose* ditunjukkan pada Kode Sumber 5.6.

Kode Sumber 5.6 Implementasi Perhitungan Matriks *Moore – Penrose*

```
Algoritme 6: Fungsi Perhitungan Matriks Moore - Penrose
1   double[][] moorePenrose(double[][] h) {
2       double[][] hT = transposeMatrix(h);
3       double[][] hTH = new double[hT.length][h[0].length];
4
5       for (int i = 0; i < hT.length; i++) {
```

```
6           for (int j = 0; j < h[0].length; j++) {
7               for (int k = 0; k < hT[0].length; k++) {
8                   hTH[i][j] += hT[i][k] * h[k][j];
9               }
10          }
11      }
12
13      double det = (hTH[0][0] * hTH[1][1]) - (hTH[0][1] * hTH[1][0]);
14      double temp = hTH[0][0];
15      hTH[0][0] = hTH[1][1];
16      hTH[1][1] = temp;
17
18      hTH[0][1] = -hTH[0][1];
19      hTH[1][0] = -hTH[1][0];
20
21      for (int i = 0; i < hTH.length; ++i) {
22          for (int j = 0; j < hTH[0].length; ++j) {
23              hTH[i][j] = hTH[i][j] / det;
24          }
25      }
26
27      double[][] hDagger = new double[hTH.length][hT[0].length];
28      for (int i = 0; i < hTH.length; i++) {
29          for (int j = 0; j < hT[0].length; j++) {
30              for (int k = 0; k < hTH[0].length; k++) {
31                  hDagger[i][j] += hTH[i][k] * hT[k][j];
32              }
33          }
34      }
35      return hDagger;
36  }
```

Penjelasan Kode Sumber 5.6 tentang implementasi perhitungan matriks *Moore – Penrose* adalah sebagai berikut:

Baris 2 : Inisialisasi variabel *hT* yang berisi hasil *transpose* variabel *h* yang diproses menggunakan method *transposeMatriks().*

Baris 3 : Deklatasi variabel *hTH* untuk menyimpan nilai.

Baris 5 s.d. 11 : Proses perhitungan perkalian dua matriks untuk dan disimpan dalam variabel *hTH*.

Baris 13 s.d. 19 : Proses perhitungan mencari nilai determinan.

Baris 21 s.d. 25 : Proses perhitungan mencari nilai *invers*.

Baris 27 s.d. 34 : Proses perhitungan mencari nilai matriks *Moore – Penrose*.

Baris 35 : Mengembalikan nilai dari variabel *hDagger*.

5.2.6 Implementasi Perhitungan *Output Weight*

Proses mencari nilai *output weight* yaitu dengan mengalikan matriks *Moore – Penrose* dan matriks target. Implementasi perhitungan *output weight* dapat dilihat pada Kode Sumber 5.7.

Kode Sumber 5.7 Implementasi Perhitungan *Output Weight*

```
Algoritme 7: Fungsi Perhitungan Output Weight
1   double[][] beta(double[][] hD, double[] t) {
2       double[][] targetTraining = konversiKelas(t);
3       double[][] b = new double[hD.length][targetTraining[0].length];
```

```
4
5       for (int i = 0; i < hD.length; i++) {
6           for (int j = 0; j < targetTraining[0].length; j++) {
7               for (int k = 0; k < hD[0].length; k++) {
8                   b[i][j] += hD[i][k] * targetTraining[k][j];
9               }
10          }
11      }
12      return b;
13  }
```

Penjelasan Kode Sumber 5.7 tentang implementasi perhitungan *output weight* adalah sebagai berikut:

Baris 2 : Inisialisasi variabel *targetTraining* berisi hasil dari perhitungan *method konversiKelas().*

Baris 3 : Deklarasi variabel *b*.

Baris 5 s.d. 10 : Proses perhitungan perkalian dua matriks.

Baris 11 : Mengembalikan nilai dari variabel b.

5.2.7 Implementasi Perhitungan *Output Layer*

Perhitungan *output layer* yaitu dengan mengalikan nilai *output hidden neuron* teraktivasi dan *output weight*. Implementasi perhitungan *output layer* ditunjukkan pada Kode Sumber 5.8.

Kode Sumber 5.8 Implementasi Perhitungan *Output Layer*

```
Algoritme 8: Fungsi Perhitungan Output Layer
1   double[][] outputLayer(double[][] h, double[][] b) {
2       double[][] y = new double[h.length][b[0].length];
3
4       for (int i = 0; i < h.length; i++) {
5           for (int j = 0; j < b[0].length; j++) {
6               for (int k = 0; k < h[0].length; k++) {
7                   y[i][j] += h[i][k] * b[k][j];
8               }
9           }
10      }
11      return y;
12  }
```

Penjelasan Kode Sumber 5.8 tentang implementasi perhitungan *output layer* adalah sebagai berikut:

Baris 2 : Deklarasi variabel *y* yang akan digunakan untuk menyimpan nilai perhitungan.

Baris 4 s.d. 10 : Proses perhitungan perkalian dua matriks kemudian hasilnya disimpan oleh variabel *y*.

Baris 11 : Mengembalikan nilai dari variabel *y*.

5.2.8 Implementasi Perhitungan Hasil Prediksi

Untuk mendapatkan kelas hasil prediksi, maka dilakukan rekonversi nilai dari hasil *output layer*. Implementasi perhitungan hasil prediksi ditunjukkan pada Kode Sumber 5.9.

Kode Sumber 5.9 Implementasi Perhitungan Hasil Prediksi

```
Algoritme 9: Fungsi Perhitungan Hasil Prediksi
1   int[] hasilPrediksi(double[][] outputLayer) {
2       double max;
3       double maxArray[] = new double[outputLayer.length];
4       double[][] hasilPrediksi = new double[outputLayer.length]
                                       [outputLayer[0].length];
5
6       for (int i = 0; i < outputLayer.length; i++) {
7           max = -99999;
8           for (int j = 0; j < outputLayer[0].length; j++) {
9               if (outputLayer[i][j] > max) {
10                  max = outputLayer[i][j];
11              }
12          }
13          maxArray[i] = max;
14      }
15
16      int[] kelas = new int[outputLayer.length];
17
18      for (int i = 0; i < outputLayer.length; i++) {
19          for (int j = 0; j < outputLayer[0].length; j++) {
20              if (outputLayer[i][j] == maxArray[i]) {
21                  kelas[i] = j + 1;
22              }
23          }
24      }
25      return kelas;
26  }
```

Penjelasan Kode Sumber 5.9 tentang implementasi perhitungan hasil prediksi adalah sebagai berikut:

Baris 2 s.d. 4 : Deklarasi variabel-variabel yang dibutuhkan dalam perhitungan.

Baris 6 s.d. 14 : Proses perhitungan untuk mencari kelas prediksi suatu data.

Baris 16 : Deklarasi variabel *kelas*.

Baris 18 s.d. 24 : Proses konversi nilai prediksi dari bilangan yang memiliki nilai -1 dan 1 menjadi kelas 1, 2, dan seterusnya.

Baris 25 : Mengembalikan nilai variabel *kelas*.

5.2.9 Implementasi Proses Evaluasi

Proses evaluasi dilakukan dengan cara mencari nilai *fitness* dari setiap individu. Nilai *fitness* didapatkan dari hasil akurasi yang didapatkan, sehingga implementasinya adalah seperti pada Kode Sumber 5.10.

Kode Sumber 5.10 Implementasi Proses Evaluasi

```
Algoritme 10: Fungsi Perhitungan Proses Evaluasi
1   double akurasi(int x, int[] hasil) {
2       int prediksiBenar = 0;
3       for (int i = 0; i < hasil.length; i++) {
4           if (kelasDataUji[i] == hasil[i]) {
5               prediksiBenar++;
6           }
7       }
8       return (((double) prediksiBenar / kelasDataUji.length)* 100);
9   }
```

Penjelasan Kode Sumber 5.10 tentang proses perhitungan evaluasi untuk mencari nilai *fitness* (akurasi) adalah sebagai berikut:

Baris 2 : Inisialisasi variabel *prediksiBenar* dengan nilai 0.

Baris 3 s.d. 7 : Proses pengecekan antara kelas aktual data uji dan kelas hasil prediksi.

Baris 8 : Mengembalikan nilai perhitungan akurasi.

5.2.10 Implementasi Perhitungan Proses Seleksi

Pada proses seleksi, dilakukan pengambilan individu yang memiliki nilai *fitness* terbesar, dan diambil sebanyak *popsize*. Implementasi proses seleksi ditunjukkan pada Kode Sumber 5.11.

Kode Sumber 5.11 Implementasi Perhitungan Proses Seleksi

```
Algoritme 11: Fungsi Perhitungan Proses Seleksi
1   double[][] seleksi(double[][] fitness) {
2       double temp[] = new double[fitness.length];
3
4       for (int i = 0; i < fitness.length; i++) {
5           for (int j = i + 1; j < fitness.length; j++) {
6               if (fitness[i][8] < fitness[j][8]) {
7                   temp = fitness[i];
8                   fitness[i] = fitness[j];
9                   fitness[j] = temp;
10              }
11          }
12      }
13
14      double[][] individuSeleksi = new double[popsize][gen];
15      for (int i = 0; i < popsize; i++) {
16          for (int j = 0; j < gen; j++) {
17              individuSeleksi[i][j] = fitness[i][j];
18          }
19      }
20      return individuSeleksi;
21  }
```

Penjelasan Kode Sumber 5.11 tentang implementasi proses seleksi adalah sebagai berikut:

Baris 2 : Deklarasi variabel *temp* yang digunakan untuk menyimpan nilai variabel *fitness* sementara.

Baris 4 s.d. 12 : Proses mengurutkan individu berdasarkan nilai *fitness* secara *descending*.

Baris 14 : Deklarasi variabel *individuSeleksi* yang digunakan untuk menampung nilai individu hasil seleksi.

Baris 15 s.d. 19 : Proses pengambilan individu sebanyak *popsize* dan disimpan pada variabel *individuSeleksi*.

Baris 20 : Mengembalikan nilai dari variabel *individuSeleksi.*

BAB 6 PENGUJIAN DAN ANALISIS

Bab ini membahas tentang proses pengujian dan analisis dari sistem yang dibangun menggunakan metode *Extreme Learning Machine* dan Algoritme Genetika. Proses pengujian pada penelitian ini meliputi pengujian konvergensi, ukuran populasi, pengaruh kombinasi nilai *crossover rate* dan *mutation rate*, pengaruh jumlah *hidden neuron* terhadap akurasi dan waktu komputasi, dan analisis global. Skenario pengujian dilakukan sesuai dengan perancangan pengujian yang telah dibuat pada tahapan sebelumnya dengan perbandingan data latih:data uji yang digunakan dalam pengujian adalah 70%:30%.

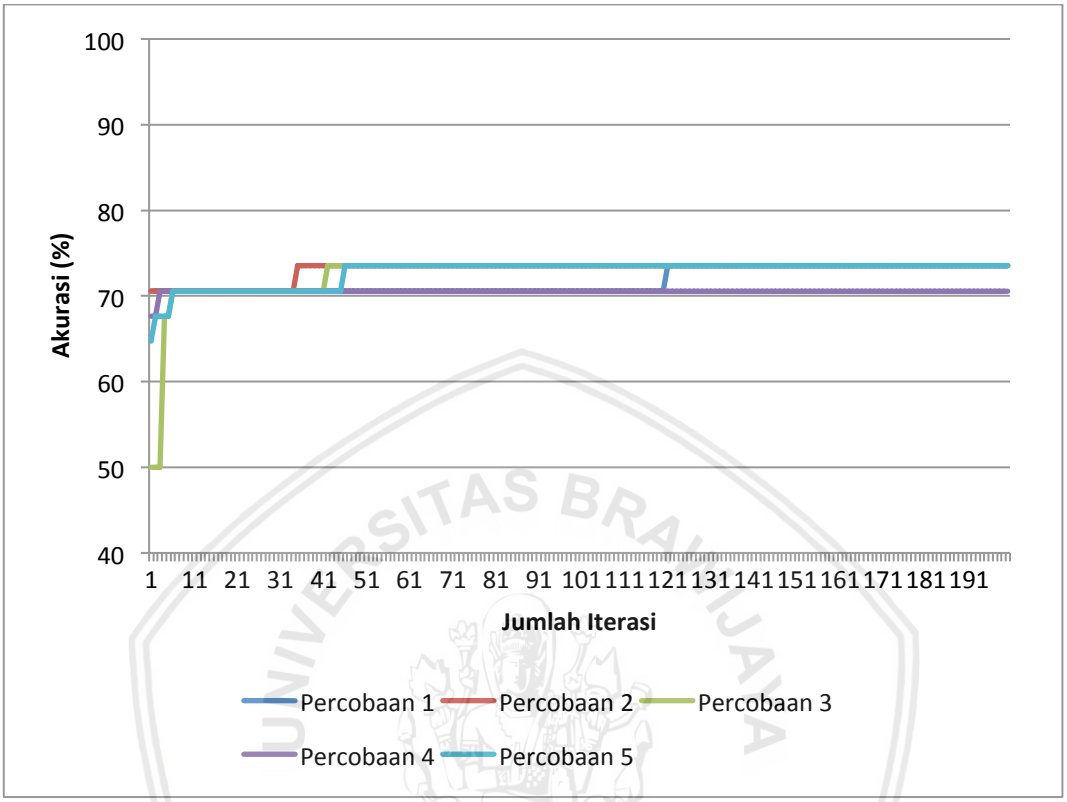
6.1 Pengujian Konvergensi

Pengujian konvergensi digunakan untuk mengetahui pada iterasi berapa nilai *fitness* konvergen. Karena nilai *fitness* yang digunakan adalah nilai akurasi dari klasifikasi yang dilakukan menggunakan metode ELM, maka setiap percobaan akan dilihat dari nilai akurasinya. Pengujian dilakukan dengan melakukan iterasi sebanyak 200 menggunakan nilai *popsize* = 10, *cr* = 0.5, *mr* = 0.5, dan *hidden neuron* sebanyak 2. Pengujian ini dilakukan sebanyak 5 kali menggunakan parameter yang sama. Hasil dari pengujian ditunjukkan pada Tabel 6.1 (Selengkapnya dapat dilihat pada Lampiran).

Tabel 6.1 Hasil Pengujian Konvergensi

Iterasi ke-	Akurasi (%)				
	Percobaan ke-				
	1	2	3	4	5
1	70.5882	70.5882	50.0000	67.6471	64.7059
2	70.5882	70.5882	50.0000	67.6471	67.6471
3	70.5882	70.5882	50.0000	70.5882	67.6471
4	70.5882	70.5882	70.5882	70.5882	67.6471
5	70.5882	70.5882	70.5882	70.5882	67.6471
...	...	...	...	...	...
196	73.5294	73.5294	73.5294	70.5882	73.5294
197	73.5294	73.5294	73.5294	70.5882	73.5294
198	73.5294	73.5294	73.5294	70.5882	73.5294
199	73.5294	73.5294	73.5294	70.5882	73.5294
200	73.5294	73.5294	73.5294	70.5882	73.5294

Dari hasil yang didapatkan seperti yang terdapat pada Tabel 6.1, maka dapat digambarkan sebuah grafik hasil pengujian konvergensi yang dapat dilihat pada Gambar 6.1.



Gambar 6.1 Grafik Pengujian Konvergensi

Berdasarkan hasil pengujian konvergensi yang telah dilakukan, pada Gambar 6.1 dapat dilihat bahwa sistem mulai menunjukkan konvergensi pada iterasi ke-5, tetapi belum mencapai kestabilan. Kemudian pada iterasi ke-6, dari 5 percobaan tersebut menunjukkan konvergensi pada nilai 70.5882%. Nilai tersebut tidak berubah atau stagnan hingga iterasi ke-34, kemudian kembali naik dan stabil pada iterasi ke-121. Namun karena waktu komputasi yang dihasilkan pada iterasi ke-121 tiga kali lebih lama dari pada waktu komputasi yang dibutuhkan pada iterasi ke-34, maka diambil iterasi sebanyak 34 untuk digunakan pada pengujian berikutnya.

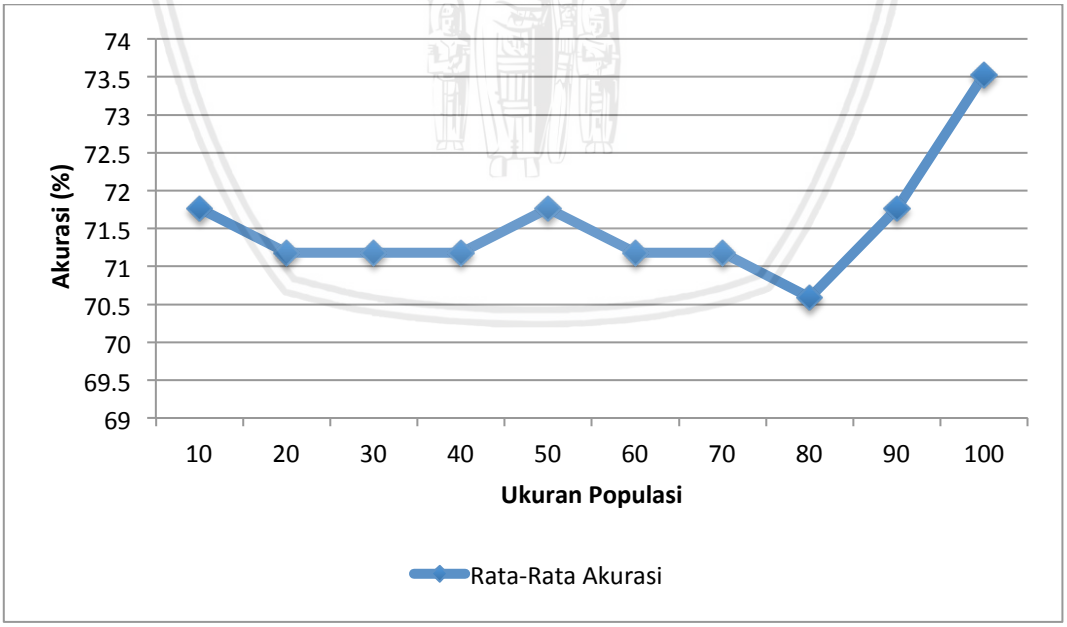
6.2 Pengujian dan Analisis Pengujian Ukuran Populasi

Pengujian ini digunakan untuk mengetahui ukuran populasi terbaik untuk menghasilkan solusi terbaik. Pengujian dilakukan dengan mengubah ukuran populasi dari 10 hingga 100 dengan interval 10. Parameter lain yang digunakan dalam pengujian yaitu *cr* = 0.5, *mr* = 0.5, *hidden neuron* = 2, dan jumlah iterasi = 34. Nilai 34 untuk iterasi diambil dari nilai iterasi pada pengujian sebelumnya. Hasil dari pengujian ukuran populasi ditunjukkan pada Tabel 6.2.

Tabel 6.2 Hasil Pengujian Ukuran Populasi

Ukuran Populasi	Akurasi					Rata-Rata Akurasi (%)
	Percobaan ke-					
	1	2	3	4	5	
10	73.5294	73.5294	70.5882	70.5882	70.5882	71.7647
20	70.5882	73.5294	70.5882	70.5882	70.5882	71.1764
30	70.5882	70.5882	73.5294	70.5882	70.5882	71.1764
40	70.5882	70.5882	70.5882	70.5882	73.5294	71.1764
50	70.5882	73.5294	73.5294	70.5882	70.5882	71.7647
60	73.5294	70.5882	70.5882	70.5882	70.5882	71.1764
70	70.5882	73.5294	70.5882	70.5882	70.5882	71.1764
80	70.5882	70.5882	70.5882	70.5882	70.5882	70.5882
90	70.5882	73.5294	70.5882	73.5294	70.5882	71.7647
100	**73.5294**	**73.5294**	**73.5294**	**73.5294**	**73.5294**	**73.5294**

Dari Tabel 6.2 dapat digambarkan sebuah grafik hasil pengujian ukuran populasi yang dapat dilihat pada Gambar 6.2.



Gambar 6.2 Grafik Pengujian Ukuran Populasi

Berdasarkan grafik pada Gambar 6.2 dapat dilihat bahwa ukuran populasi 100 menghasilkan rata-rata akurasi terbaik, yaitu 73.5293%. Sedangkan pada ukuran populasi 80 menghasilkan rata-rata akurasi terendah, yaitu 70.5882%. Grafik di atas juga menunjukkan bahwa ukuran populasi memengaruhi akurasi sistem,

namun semakin banyak populasi tidak selalu membuat akurasi semakin baik karena nilai *random* kromosom yang digunakan bertepatan memiliki nilai *fitness* yang rendah. Hal tersebut dapat dilihat saat ukuran populasi bernilai 10 hingga 20 serta 50 hingga 80.

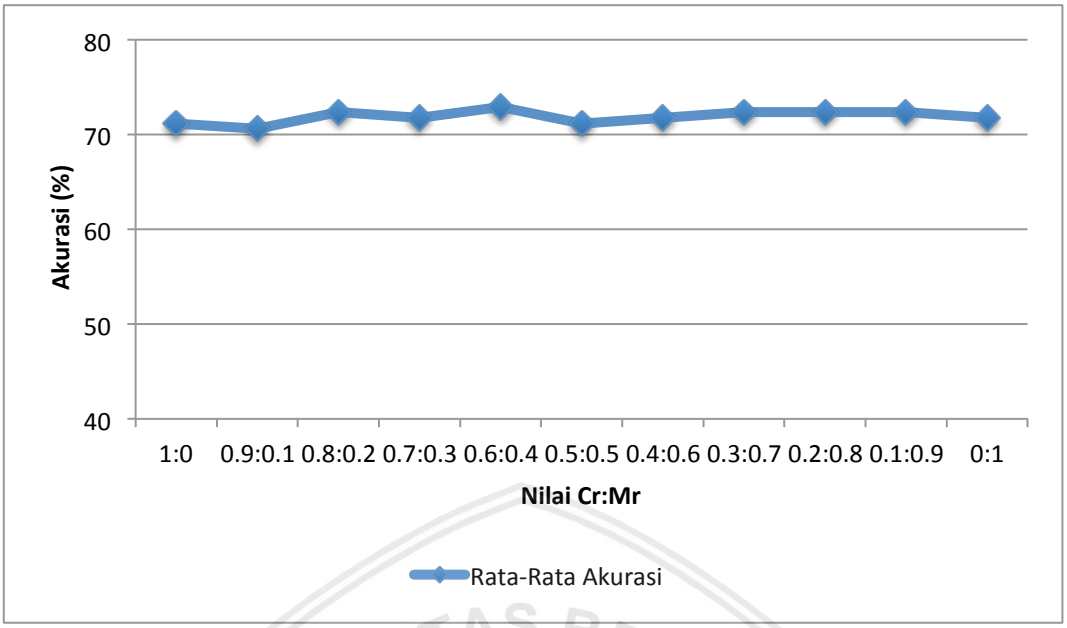
6.3 Pengujian dan Analisis Pengujian Pengaruh Kombinasi Nilai *Crossover Rate* dan *Mutation Rate*

Pengujian pengaruh kombinasi nilai *crossover rate* (*cr*) dan *mutation rate* (*mr*) digunakan untuk mengetahui apakah kedua nilai tersebut memengaruhi hasil perhitungan, serta mencari kombinasi nilai *cr* dan *mr* terbaik guna mendapatkan solusi terbaik. Pengujian dilakukan dengan mengubah kombinasi nilai *cr*:*mr* dari 1:0 hingga 0:1 dengan interval 0.1. Setiap kombinasi nilai dilakukan percobaan sebanyak 5 kali untuk mendapatkan rata-rata akurasinya. Parameter lain yang digunakan dalam pengujian adalah nilai iterasi = 34, *popsize* = 100 yang diambil dari nilai *popsize* terbaik dari pengujian ukuran populasi, dan *hidden neuron* = 2. Hasil pengujian pengaruh kombinasi nilai *cr* dan *mr* dapat dilihat pada Tabel 6.3.

Tabel 6.3 Hasil Pengujian Pengaruh Kombinasi Nilai *Cr* dan *Mr*

Nilai Kombinasi		Akurasi (%)					Rata-Rata Akurasi (%)
		Percobaan ke-					
cr	*mr*	1	2	3	4	5	
1	0	70.5882	70.5882	70.5882	70.5882	70.5882	70.5882
0.9	0.1	70.5882	70.5882	70.5882	73.5294	70.5882	71.1764
0.8	0.2	73.5294	70.5882	70.5882	73.5294	73.5294	72.3529
0.7	0.3	73.5294	73.5294	70.5882	70.5882	70.5882	71.7647
0.6	**0.4**	**73.5294**	**73.5294**	**73.5294**	**70.5882**	**73.5294**	**72.9412**
0.5	0.5	70.5882	70.5882	70.5882	70.5882	73.5294	71.1764
0.4	0.6	70.5882	73.5294	70.5882	70.5882	73.5294	71.7647
0.3	0.7	70.5882	73.5294	73.5294	73.5294	70.5882	72.3529
0.2	0.8	73.5294	70.5882	70.5882	73.5294	73.5294	72.3529
0.1	0.9	73.5294	73.5294	70.5882	73.5294	73.5294	72.3529
0	1	70.5882	70.5882	70.5882	70.5882	70.5882	70.5882

Berdasarkan hasil pengujian pengaruh kombinasi nilai *cr* dan *mr* seperti yang ditunjukkan pada Tabel 6.3, dihasilkan rata-rata nilai akurasi untuk masing-masing percobaan. Grafik hasil pengujian pengaruh kombinasi nilai *cr* dan *mr* ditunjukkan pada Gambar 6.3.



Gambar 6.3 Grafik Pengujian Pengaruh Kombinasi Nilai *Cr* dan *Mr*

Berdasarkan hasil pengujian yang dilakukan, dapat dilihat bahwa kombinasi nilai *cr* dan *mr* memengaruhi akurasi sistem meskipun tidak memberikan perbedaan yang signifikan. Rata-rata akurasi terbaik dihasilkan oleh kombinasi nilai *cr*:*mr* sebesar 0.6:0.4 yakni sebesar 72.9412%, dan rata-rata akurasi terendah dihasilkan oleh kombinasi nilai *cr*:*mr* sebesar 1:0 dan 0:1 dengan rata-rata akurasi sebesar 70.5882%. Pengujian ini dapat disimpulkan bahwa nilai *cr* dan *mr* yang terlalu besar atau terlalu kecil cenderung menghasilkan akurasi yang rendah. Nilai *cr* yang terlalu kecil dan nilai *mr* yang terlalu besar mengakibatkan Algoritme Genetika tidak mampu memperlebar area pencarian. Sebaliknya, nilai *cr* yang terlalu besar dan *mr* yang terlalu kecil akan membuat *offspring* kemiliki kemiripan tinggi dengan *parent*-nya sehingga menyebabkan konvergensi dini (Mahmudy, Marian, & Luong, 2014).

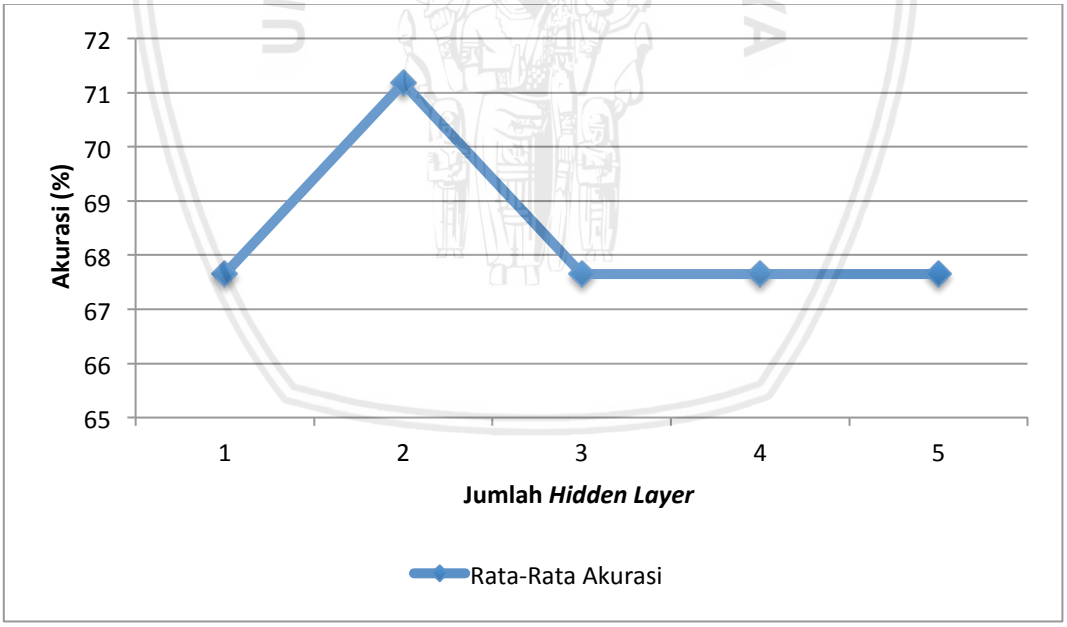
6.4 Pengujian Jumlah *Hidden Neuron*

Pengujian jumlah *hidden neuron* terhadap waktu digunakan untuk mengetahui bagaimana pengaruh jumlah nilai *hidden neuron* terhadap waktu komputasi serta akurasi yang didapatkan. Pengujian dilakukan dengan mengubah nilai jumlah *hidden neuron* dari 1 hingga 5. Pengujian masing-masing jumlah *hidden neuron* dilakukan sebanyak 5 kali, dengan parameter pendukung lain nilai *popsize* = 100, iterasi = 9, serta kombinasi nilai *cr:mr* = 0.6:0.4. Nilai dari parameter-parameter tersebut merupakan nilai terbaik dari masing-masing pengujian yang telah dilakukan. Hasil pengujian jumlah *hidden neuron* terhadap akurasi dan waktu ditunjukkan pada Tabel 6.4.

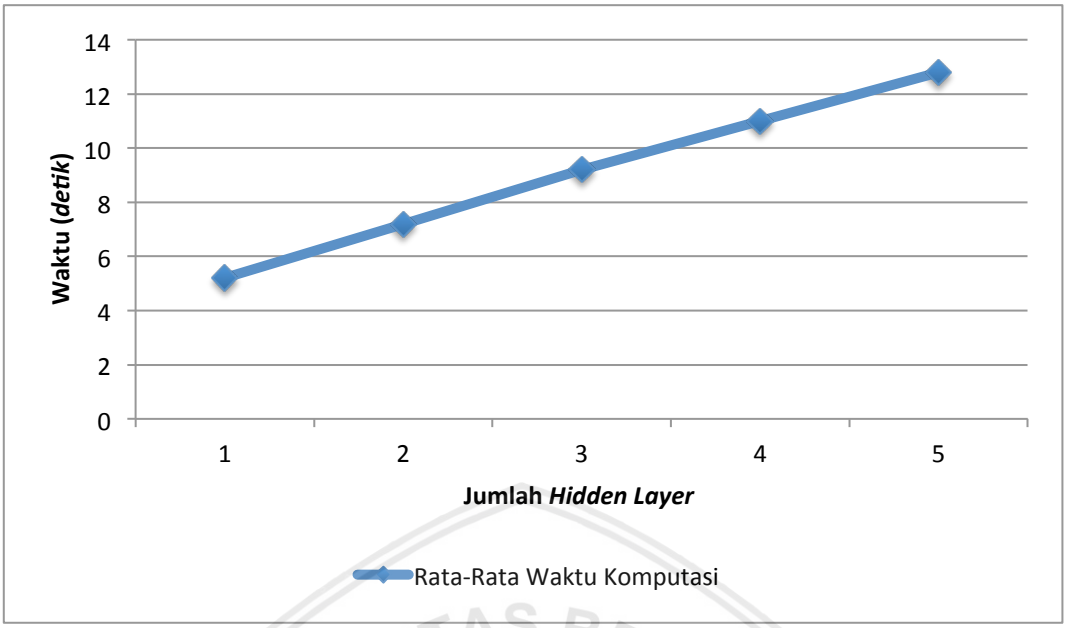
Tabel 6.4 Hasil Pengujian Jumlah *Hidden Neuron*

Percobaan ke-	Jumlah *Hidden Neuron*									
	1		2		3		4		5	
	A	W	A	W	A	W	A	W	A	W
1	67.65	6	70.59	7	67.65	10	67.65	10	67.65	13
2	67.65	5	70.59	7	67.65	9	67.65	12	67.65	13
3	67.65	5	73.53	7	67.65	10	67.65	11	67.65	13
4	67.65	5	70.59	7	67.65	8	67.65	11	67.65	13
5	67.65	5	70.59	8	67.65	9	67.65	11	67.65	12
Rata-Rata	67.65	5.2	71.18	7.2	67.65	9.2	67.65	11	67.65	12.8

Berdasarkan hasil pengujian jumlah *hidden neuron* seperti yang ditunjukkan pada Tabel 6.4, dihasilkan rata-rata nilai akurasi dan waktu komputasi untuk masing-masing percobaan. Grafik hasil pengujian jumlah *hidden neuron* terhadap nilai akurasi ditunjukkan pada Gambar 6.4 dan grafik hasil pengujian jumlah *hidden neuron* terhadap nilai waktu komputasi ditunjukkan pada Gambar 6.5.



Gambar 6.4 Hasil Pengujian Jumlah *Hidden Neuron* Terhadap Akurasi



Gambar 6.5 Grafik Pengujian Jumlah *Hidden Neuron* Terhadap Waktu Komputasi

Dari percobaan yang telah dilakukan, dapat disimpulkan bahwa jumlah *hidden neuron* memengaruhi waktu komputasi serta akurasi. Grafik pada Gambar 6.4 menunjukkan bahwa semakin banyak *hidden neuron* yang digunakan tidak selalu membuat nilai akurasi semakin baik. Hal tersebut dapat dilihat dari nilai rata-rata akurasi tertinggi diperoleh saat *hidden neuron* bernilai 2. Kemudian untuk waktu komputasi, grafik pada Gambar 6.5 menunjukkan bahwa semakin banyak jumlah *hidden neuron*, semakin lama juga waktu komputasi yang dibutuhkan oleh sistem. Hal tersebut dapat dilihat dari waktu terendah dihasilkan saat *hidden neuron* bernilai 1, dan waktu tertinggi ditunjukkan pada saat *hidden neuron* bernilai 5.

6.5 Analisis Global dari Hasil Pengujian

Pada penelitian ini dilakukan 4 macam pengujian terkait dengan metode ELM dan Algoritme Genetika. Pengujian yang dilakukan meliputi pengujian konvergensi, ukuran populasi, pengaruh kombinasi nilai *cr* dan *mr*, dan jumlah *hidden neuron*. Pengujian pertama yaitu pengujian konvergensi yang didapatkan hasil nilai konvergen pada iterasi ke-34. Pengujian kedua yaitu pengujian ukuran populasi dengan mengubah nilai populasi dari 10 sampai 100 dengan interval 10. Pengujian tersebut mendapatkan hasil bahwa ukuran populasi sebanyak 100 mampu menghasilkan rata-rata akurasi terbaik sebesar 73.5294%. Pengujian ketiga yaitu pengujian pengaruh kombinasi nilai *cr* dan *mr*. Hasil dari pengujian tersebut yaitu kombinasi nilai *cr*:*mr* sebesar 0.6:0.4 mampu memberikan rata-rata akurasi terbaik, yaitu 72.9412%. Pengujian keempat yaitu pengujian jumlah *hidden neuron* yang menghasilkan jumlah hidden neuron 2 mampu memberikan rata-rata akurasi terbaik sebesar 71.18%.

Berdasarkan hasil pengujian yang menghasilkan parameter optimal, maka dilakukan pengujian pada metode ELM dan ELM – Algoritme Genetika (AG) untuk membandingkan hasil akurasi pada metode tersebut. Perbandingan metode ELM dan ELM – AG ditunjukkan pada Tabel 6.5.

Tabel 6.5 Hasil Perbandingan Akurasi pada ELM dan ELM - AG

Percobaan ke-	Akurasi (%)	
	ELM	ELM – AG
1	67.6471	73.5294
2	67.6471	73.5294
3	67.6471	70.5882
4	67.6471	73.5294
5	67.6471	70.5882
Rata-rata	67.6471	72.3529

Setelah dilakukan percobaan sebanyak 5 kali, maka didapatkan rata-rata akurasi perhitungan menggunakan ELM sebesar 67.6471%, dan ELM – AG sebesar 72.3529%. Berdasarkan hasil tersebut, dapat disimpulkan bahwa Algoritme Genetika mampu mencari bobot yang optimal untuk ELM sehingga dapat meningkatkan akurasi sebesar 4.7058%. Dengan rata-rata akurasi 72.3529% pada ELM – AG juga belum maksimal, dikarenakan data yang digunakan tidak seimbang sehingga terdapat kelas yang tidak dapat dikenali polanya.

BAB 7 PENUTUP

Bagian penutup membahas mengenai kesimpulan dari penelitian yang telah dilakukan serta saran terhadap penelitian sejenis atau pengembangan selanjutnya dari penelitian yang telah dilakukan ini.

7.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis mengenai klasifikasi status gizi balita menggunakan *Extreme Learning Machine* dan Algoritme Genetika, maka didapatkan kesimpulan sebagai berikut:

1. Berdasarkan hasil pengujian yang telah dilakukan, parameter optimal yang didapatkan untuk mengklasifikasikan status gizi pada balita yaitu parameter *popsize* bernilai 100, parameter iterasi sebanyak 34, parameter nilai *cr* sebesar 0.6, parameter nilai *mr* sebesar 0.4, dan parameter *hidden neuron* bernilai 2.
2. Rata-rata akurasi terbaik dengan menggunakan parameter optimal pada penelitian ini mencapai 72.3529%, meningkat 4.7058% dari rata-rata akurasi yang dihasilkan oleh ELM yakni sebesar 67.6471%.

7.2 Saran

Adapun saran yang dapat digunakan untuk penelitian selanjutnya adalah sebagai berikut:

1. Penelitian ini hanya menggunakan bobot (*input weight*) yang kemudian dilakukan optimasi sebagai parameter pada ELM, penelitian selanjutnya dapat menambahkan bias sebagai parameter lain pada ELM serta juga dapat melakukan optimasi terhadap nilai biasnya.
2. Karena data yang digunakan dalam penelitian tidak seimbang, maka dapat menyeimbangkan data yang digunakan dengan mengimplementasikan metode *data mining* seperti *Synthetic Minority Oversampling Technique* (SMOTE) sehingga akurasi juga dapat meningkat.

DAFTAR PUSTAKA

Ali, M. H., Zolkipli, M. F., Mohammed, M. A., & Jaber, M. M., 2017. Enhance of Extreme Learning Machine-Genetic Algorithm Hybrid Based on Intrusion Detection System. *Journal of Engineering and Applied Sciences*, 12(16), 4180–4185.

Amalia, H., 2015. Optimasi Neural Network Menggunakan Genetic Algorithm Untuk Prediksi Penyakit Diabetes. *Paradigma*, XVII(2), 69–78.

Anggraeni, R., & Indrarti, A., 2010. Klasifikasi Status Gizi Balita Berdasarkan Indeks Antropometri Menggunakan Jaringan Syaraf Tiruan. *Seminar Nasional Aplikasi Teknologi Informasi* (SNATI), 2008, 14–18.

Ariestyani, M. C., Adikara, P. P., & Perdana, R. S., 2018. Klasifikasi Penyimpangan Tumbuh Kembang Anak Menggunakan Metode Extreme Learning Machine (ELM). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, 2(4), 1620–1629.

Chandra, R. A., Santoso, E., & Adinugroho, S., 2018. Optimasi Metode Extreme Learning Machine Dalam Penentuan Kualitas Air Sungai Menggunakan Algoritme Genetika. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, 2(10), 3265–3273.

Cholissodin, I., & Riyandani, E., 2016. Analisis Big Data (Teori & Aplikasi). [pdf] <Tersedia di: http://bit.ly/2x8ta9S>

De Onis, M., Blössner, M., & Borghi, E., 2012. Prevalence and Trends of Stunting Among Pre-School Children, 1990-2020. *Public Health Nutrition*, 15(1), 142–148.

Fadilla, I., Adikara, P. P., & Perdana, R. S., 2018. Klasifikasi Penyakit Chronic Kidney Disease (CKD) Dengan Menggunakan Metode Extreme Learning Machine (ELM), *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, 2(10), 3397–3405.

Hidayat, R., & Suprapto., 2012. Meminimalisasi Nilai Error Peramalan dengan Algoritma Extreme Learning Machine. *Jurnal Optimasi Sistem Industri*, 11(1), 187–192.

Holland, J. H. (2005). Genetic Algorithms, 12–15.

Huang, G. Bin, Zhu, Q. Y., & Siew, C. K., 2006. Extreme Learning Machine: Theory and Applications. *Neurocomputing*, 70(1–3), 489–501.

Irsyadi, F. Y. Al, & Fathina, H., 2015. Klasifikasi Status Gizi Balita Jenis Kelamin Laki-laki Menggunakan Jaringan Syaraf Tiruan. *Khazanah Informatika: Jurnal Ilmu Komputer Dan Informatika*, 1(1), 16–22.

Mahmudy, W. F., 2015. Modul Kuliah Semester Ganjil 2015-2016: Dasar-Dasar Algoritma Evolusi. [pdf] <Tersedia di: wayanfm.lecture.ub.ac.id/>

Mahmudy, W. F., Marian, R. M., & Luong, L. H. S., 2014. Hybrid Genetic Algorithms for Part Type Selection and Machine Loading Problems with Alternative Production Plans in Flexible Manufacturing System. *ECTI Transactions on Computer and Information Technology*, *8*(1), 80–93.

Martiana, E., 2015. Aplikasi Untuk Diagnosa Gizi Pada Balita Serta Kandungan Kalori yang Diperlukan Guna Mendapatkan Gizi Seimbang Menggunakan Metode Fuzzy Sugeno. [pdf] <Tersedia di: researchgate.net/publication>.

Oktaviasari, D. I., & Muniroh, L., 2009. Hubungan Antara Besar Pengeluaran Keluarga untuk Rokok dengan Status Gizi Balita pada Keluarga Miskin. *Journal of Empirical Research on Human Research Ethics: JERHRE*, *9*(1), 10–18.

Ramadonna, T. F., Sivia, A., & Ciksadan., 2017. Perbandingan Algoritma Genetika dan TSP Untuk Optimalisasi Jaringan Akses Fiber To The Home. *Jurnal Teknik Informatika Dan Sistem Informasi, 3(2)*, 344–353.

Yaseen, Z. M., Deo, R. C., Hilal, A., Abd, A. M., Bueno, L. C., Salcedo-Sanz, S., & Nehdi, M. L. (2018). Predicting Compressive Strength of Lightweight Foamed Concrete Using Extreme Learning Machine Model. *Advances in Engineering Software*, *115*, 112–125.